SLICING OF TESSELLATED MODELS FOR ADDITIVE MANUFACTURING BASED ON VARIABLE THICKNESS LAYERS

A Dissertation
Presented to
The Academic Faculty

By

Dongmin Han

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Mechanical Engineering

Georgia Institute of Technology

December 2019

SLICING OF TESSELLATED MODELS FOR ADDITIVE MANUFACTURING BASED ON VARIABLE THICKNESS LAYERS

Approved by:

Dr. Thomas Kurfess, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Christopher Saldana
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Katherine Fu
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Yan Wang
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Tommy Tucker
Tucker Innovations, Inc.

Date Approved: October 21, 2019

ACKNOWLEDGEMENTS

I would first like to thank my advisor, Dr. Thomas Kurfess, for his encouragement and guidance in the past four years. I would also like to thank all my committee members, Dr. Christopher Saldana, Dr. Katherine Fu, Dr. Yan Wang and Dr. Tommy Tucker for supporting me through all the research efforts I made while at Georgia Tech, especially this project. Thanks are also due to all my friends and colleagues, especially Vinh and Max, for their help and contribution to this project. Finally, I deeply grateful to my mother and to my wife Xinyu. Thank you for supporting me and accompanying me through all the good times and bad.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In contrast to machining or subtractive technologies, Additive Manufacturing (AM) is a set of technologies that fabricate a 3-D object by automatically adding material layer-by-layer. In AM systems, the Computer Aided Design (CAD) model is converted into layers in a process known as slicing. One of the limitations of AM is the geometrical inaccuracy and undesirable surface finish due to the layer-upon-layer application of material. Inclined features suffer significantly from this drawback, known as the stair-step effect. While decreasing the layer thickness can reduce the stair-step effect, the cost of considerably increasing processing time is unappealing to manufacturers.

Flat layer additive manufacturing has a number of limitations: first, the trade-off between better surface finish and printing time; second, support structure are usually needed, which causes the unwelcome surface quality on the contact areas between the part and the support structure; and third, the use of flat layer leads to the anisotropy property, which affects the strength of the final parts. To overcome the limitations present in flat layer additive manufacturing, adaptive slicing and curved-layer slicing were proposed. Some research has focused on developing the algorithms that adaptively chooses the layer thickness based on the curvature and angle along the surface. Some has developed curved layer slicing by offsetting the top surface to generate the layers. But all of these works only apply to 3D models with lots of constrains and involve certain level of manual interventions. In addition, all of these works are only aiming at 3-axis FDM machines, while the proposed slicing procedure is not only applicable to 3-axis systems, but also suitable for 5-axis.

This research proposes a new solution to slice tessellated CAD models with dynamic thickness layers. The proposed method negates the stair-step effect and provides smooth bonding between layers. It also provide the potential to be applied on 5-axis FDM machines with minimum modifications. In this procedure, the CAD models is divided into planar-curved regions and uniform slicing regions by the directions of the facet vectors.

The top surface is extracted from the curved region and the facets are offset with different distance from the top surface to create slicing layers. As a result, the dimensional accuracy is improved using fewer layers compared to uniform slicing. Hence, the proposed method can significantly save print time without compromising quality. In addition, a more generic slicing procedure will be developed by applying the method locally to individual features on a single part. The contributions of the research are as follows: first, a dynamic thickness curved layer slicing algorithm for tessellated models was developed; second, this approach was implemented on a 3-axes Fused Deposition Modeling (FDM) platform; third, the surface integrity property was improved; and fourth, a more generic slicing algorithm was developed for more complicated models.

CHAPTER 1
INTRODUCTION AND BACKGROUND

CAD has significantly improved the ability to design engineering parts due to its ease of use. Almost any geometry can be modeled by using CAD software. The capability of AM to fabricate complex forms in a single stage relaxes manufacturing constraints on the design process. In addition, the manufacturing lead time of a product can potentially be greatly reduced by using AM technology [1]. Interactions with 3D printers is through the printer's toolchain, which includes the electronics, firmware and the slicing software [2]. This workflow is shown in 1.1. For each step along the toolchain, a set of hardware and applications needs to be used to actualize the workflow.

In this simplified demonstration, the 3D model acts in the control software as the input of the system. The control application then sketches the model and provides printing settings for the user, if needed. Next, the control application sends the model and the settings to the slicer and waits for the slicer to send back the generated G-Code. Then, the control application passes on the G-Code to a specific software, called firmware, that provides the low-level control on the hardware platform. The firmware processes the G-Code and controls the movement, typically using stepper-motor-driven screws on the 3D printer, to build the 3D object. It also sends information back to the control application, e.g., positions of each axis, extruder temperature, etc.

In most AM processes, the geometric models are created using CAD software. Before printing, the models have to be oriented and positioned in the workspace of the printer and cut into layers. Furthermore, support structures must be added where appropriate, and finally a tool path must be generated for each layer. There are primarily two ways to slice CAD models. One is to approximate the 3D shape with planar triangular patches, which is called tessellation, then to slice the tessellated model into layers. The other way to slice

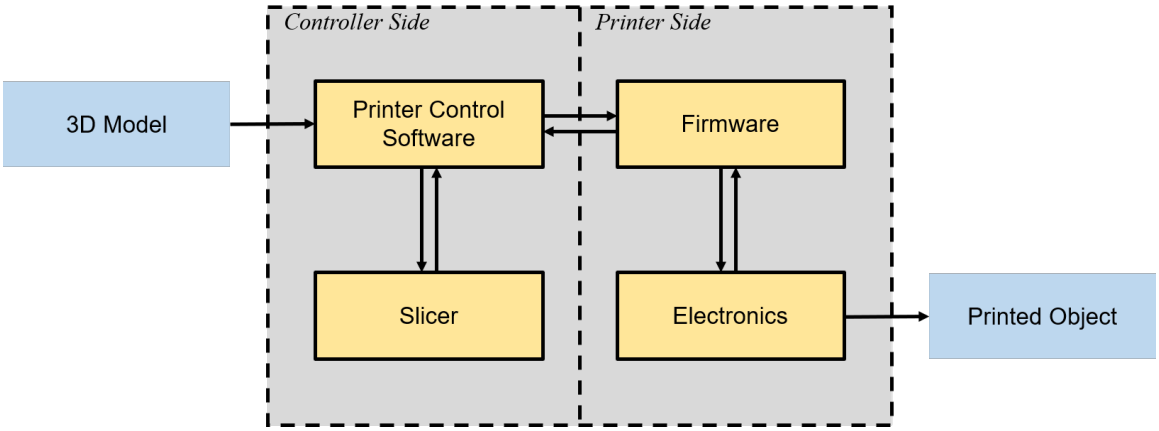


Figure 1.1: A general 3D printer toolchain.

CAD models is to directly slice the CAD models, which is known as direct slicing [3]. Tessellated models [4] are selected as the focus of this research for reasons that will be discussed in this chapter.

1.1 Thesis Outline

In this thesis, chapter 1 introduces the comprehensive background on additive manufacturing and slicing procedures. A literature review of slicing algorithms closely related to this research is also presented. The introduction covers some fundamental concepts in additive manufacturing, including STL file format, uniform slicing algorithm, G-Code and fused deposition modeling (FDM). The literature review covers the stair-step effect and existing solutions. Then, this research is introduced to overcome the limitations identified in the literature review.

Chapter 2 introduces the hardware and the software that this research is devoted to. The hardware includes the Anet A8 3D printer and its control and monitoring system hosted on a BeagleBone Black (BBB). The software includes the Marlin firmware, which drives the printer and the Cura slicing program.

Chapter 3 describes the proposed slicing process for the base case in this research. It covers the steps from the slicing algorithm, path planning to the G-Code generation.

The slicing algorithm divides the intersection of each layer into two regions, i.e., Uniform Thickness Region (UTR) and Variable Thickness Region (VTR). Then, the toolpaths generated on these two regions are connected to form a continuous toolpath for the layer. Finally, the corresponding G-Code is generated for each layer.

Chapter 4 attempts to generalize the slicing method described in chapter 3. To address the limitations of the base case algorithm, the models are further classified into two sub-cases: multiple VTRs in the slicing direction and multiple VTRs in the orthogonal slicing direction. A general case can be decomposed into a number of sub-cases.

To evaluate the performance of the variable thickness slicing algorithm, Chapter 5 compares the top surface roughness and tensile strength between the variable thickness slicing and uniform slicing algorithms.

Chapter 6 summarizes the conclusions and the contributions from this thesis. The potential future work and recommendations is also included in this chapter.

1.2 Additive Manufacturing

Additive manufacturing (AM) is also known as rapid prototyping (RP) or 3D printing. It is a technology that first developed to build prototypes from digital models. But now it is used for many more purposes, due to improvements in the build quality from the machines [1]. Therefore, more final products are made directly from AM technology. The essential principle of this technology is to build the parts by adding material layer by layer. Each layer is a cross section of the part in a certain height. The cross sections can be obtained by slicing the CAD model with specific thickness. The general process of AM technology includes a number of steps that convert the CAD model to the final part. A 3D model is designed in a CAD software. Then, it is converted to STL format. The model is sliced into layers, and a corresponding toolpath is generated for each layer. After that, the toolpath is sent to the machine, and the part is fabricated layer-by-layer. Finally, some post processing, such as removing support material, sanding, gluing may also be necessary,

depending on the material and the machine type. This general AM process is described in figure 1.2. The steps in the blue dotted box are the scope of this research.

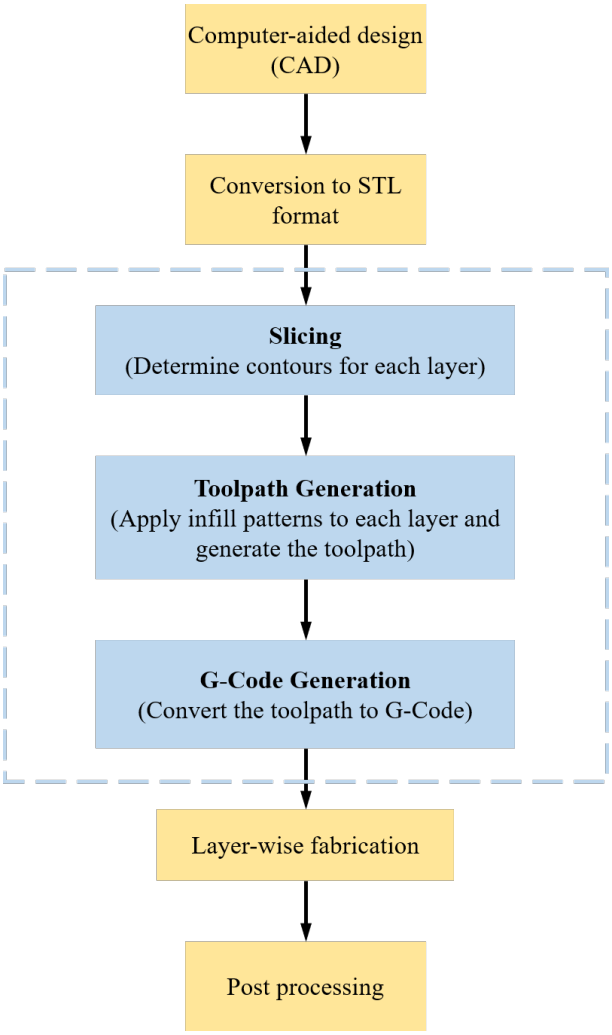


Figure 1.2: A general additive manufacturing process.

There are two types of slicing methods to slice a digital model into layers, i.e., STL-based slicing and direct slicing, based on the source data. Although direct slicing can generate more precise contours directly from the 3D model, STL-based slicing is applicable to most use cases. The STL file format is the most commonly used format, so it is a *de facto* industry standard in additive manufacturing owning to its versatility and simplicity for tessellating surfaces. Therefore, this research is focused on the STL-based slicing procedure for its universality.

1.2.1 STL File Format

In 3D printing technology, objects are built layer by layer. This process requires 2D contours to represent each layer of the object. The ideal format for such a process would be a series of polygons with height according to its z-value, or a series of meshed surfaces representing each layer. However, the object can be sliced with different layer thickness for different build speeds and precision; therefore, it is easier to represent the model in the format that allows all possible slicing techniques. An example of an STL model is shown in figure 1.3. The STL file is a standard format to encode the surface geometry of a 3D model

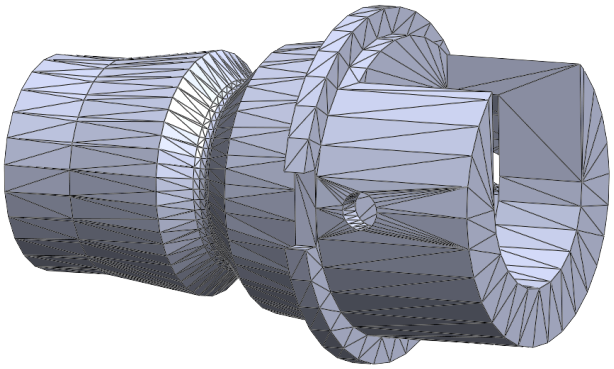


Figure 1.3: An example of STL model.

using tessellation [4]. The file format tessellates the surface with unordered triangular patches. The STL file format has two ways to store the triangular data—ASCII format and binary format—shown as below. The ASCII format starts with the name of the 3D model followed by information about the triangles representing the object. n_x, n_y, n_z is the normal to the triangle and v_1, v_2, v_3 are the vertices of the triangle. The coordinates are represented in floating point numbers using normalized scientific notation, e.g., "1.2340000e-005". In this format, some other syntaxes are allowed based on the structure of the format. (e.g., some facets can have more than one loop, or some loops can have more than three vertices,

etc.)

The size of the ASCII-format STL file can be large when the user sets a very small tolerance or the model's shape is complex. The binary format has a more compact size but lacks notation to discern individual numeric values.

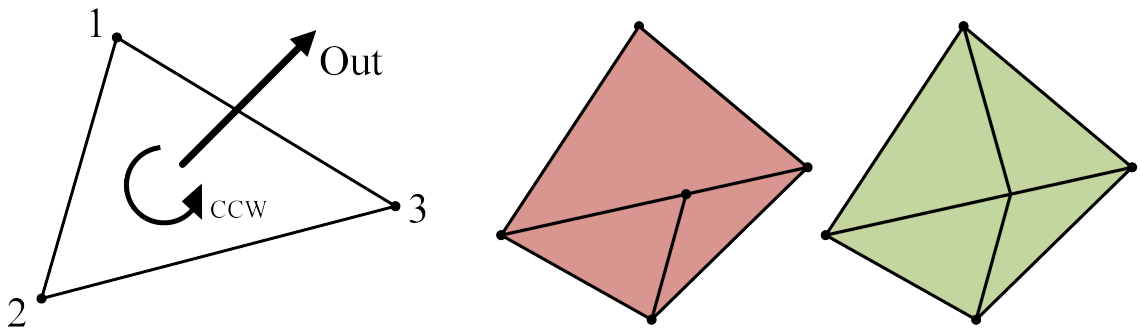


Figure 1.4: STL format rules.

Table 1.1: STL formats.

Binary Format	***ASCII Format***
UINT8[80] – Header	*solid name*
UINT32 – Number of triangles	*facet normal nx ny nz*
foreach triangle	*outer loop*
REAL32[3] – Normal vector	*vertex v1x v1y v1z*
REAL32[3] – Vertex 1	*vertex v2x v2y v2z*
REAL32[3] – Vertex 2	*vertex v3x v3y v3z*
REAL32[3] – Vertex 3	*endloop*
UINT16 – Attribute byte count	*endfacet*
End	*endsolid name*

The STL format has two rules, a facet orientation rule and a vertex-to-vertex rule. The facet orientation rule ensures that the direction of the normal of each facet is outward,

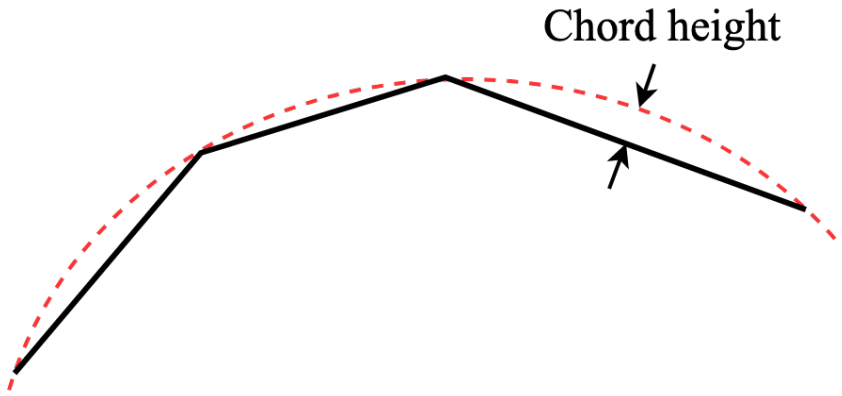
and the vertices are listed in counterclockwise order when looking at the object from the outside; this is known as the right hand rule.The vertex-to-vertex rule stipulates that each triangle must share 2 vertices with each of its adjacent triangles. These two rules are shown in figure 1.4. The left most triangle with its normal vector explains the facet orientation rule. The red patch in the middle shows a wrong mesh of the surface since facet 3 is not sharing 2 vertices with facet 1 and facet 2. The green patch shows the correct way.

Limitations of STL File

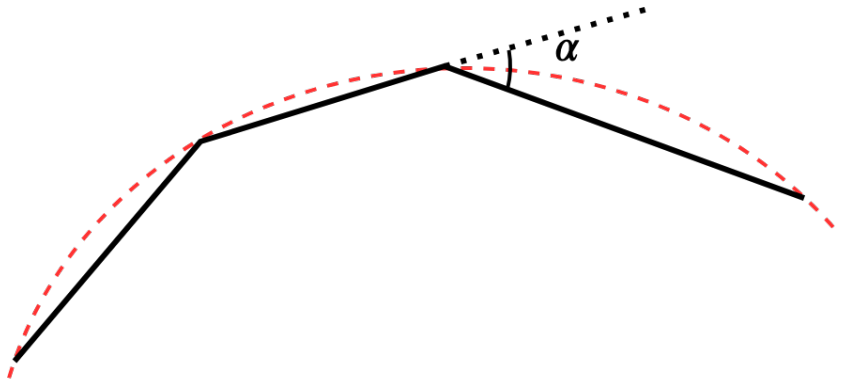
The STL format is light and simple as introduced above; however, it still has a number of limitations discussed as follows.

- Geometrical Error

  One of the biggest limitations of STL is that it defines models based on triangles. This will introduce geometrical error no matter how small the triangles are. And to obtain a better resolution of the model, a larger number of triangles would be added to the STL model. Most modern CAD software provides resolution options when exporting a STL file. The resolution is controlled by one of the two parameters, i.e., maximum chord height or angular tolerance. Figure 1.5 illustrates these two parameters. The chord height, as shown in figure 1.5a, is the distance between the surface of the original 3D model and the surface of the STL model. A smaller chord height results in more accurate surface representation. The angular tolerance, as shown in figure 1.5b, is the maximum allowed angle between the normal vectors of adjacent triangles. Using a smaller angular tolerance will help generate more accurate STL models.



(a) Chord height



(b) After STL model rebuild

Figure 1.5: Two parameters that control the resolution of STL model.

- STL File Leaking

  Although STL is the standard for 3D printing, an STL model is not always printable. STL models must be fully enclosed and manifold (i.e. each vertex must not be on any triangle edge). Also, no self-intersection is allowed either. According to the Euler's

Formula for solids, these rules can be represented in equation 1.1.

$$F + V - E = 2 \times B \tag{1.1}$$

where F is the number of faces, V is the number of vertices, E is the number of edges and B is the number of bodies. However, none of these rules is explicitly held by the STL file format itself, which can potentially cause a leaking STL file. Therefore, there are changes when the STL model is not presented correctly by the format.

- Unit Uncertainty

  The STL format doesn't explicitly define the unit of the model, and it has no mechanism for representing the scale. Therefore, confusion may be caused by using different measurements in different steps of an AM procedure. For example, if a person builds a model in imperial measurements while reading the STL in a slicer software with metric measurement, the printed part will be 25 times smaller than the designed model. This can cause unexpected consequences, especially when the person is unaware of the purpose of the part.

- Lack of Information

  The STL file can only describe the geometry of the model. There is no possible way for STL file to include other information, like color, materials, and texture. Although most of the 3D printers can only extrude one kind of material at a time, more and more 3D printers support dual extrusion, nozzle switching and material changing. For those applications which require information other than geometry, STL may not be capable of delivering all the information needed.

STL Alternatives

There are a number of alternatives to the STL format for AM. In this section, the most well-known formats are introduced and compared to STL.

- OBJ File

  The OBJ format was developed by Wavefront Technologies [5]. Similar to STL ASCII format, the OBJ format represents polygonal data in ASCII form containing vertex (v), normal vector (vn), faces (f) and texture coordinates (vp). A material library (MTL) file, which references the colors and materials, may be recalled in the OBJ file. Compared to the STL format, the OBJ format can not only encode color, texture and material information into the file, but can also handle non-triangular faces (i.e. polygons).

- AMF File

  An AMF file can represent one object containing a set of non-overlapping volumes. Like STL, each volume is represented by a set of triangular facets. AMF uses XML to encode the object information with five top level elements listed below:

  - object, specifying volumes of material with a material ID.
  - material, specifying materials corresponding to a material ID.
  - texture, specifying textures with a texture ID.
  - constellation, specifying relative pattern of objects.
  - metadata, describing any additional information of the objects.

  Compared to STL, other than material and texture information, AMF format also supports curved triangle patches, which can remarkably reduce the number of faces required to describe the surface.

Even though STL has a number of limitations, it is chosen in this research because STL is the most commonly used format in AM processes. It is supported by almost all 3D printers and CAD software. While the STL format does not support colors or material type, this research only focuses on the slicing procedure, which is independent of the aforementioned properties. There are a number of formats that can provide more comprehensive

information of the models. However, as of now, none of these alternatives have been as widely adopted as STL file. Although as 3D printing technology continues to advance, some of those STL alternatives may begin to surpass STL. But as long as those formats are triangle-based or polygon-based with extended functionalities, the proposed procedure can still be applied to those formats.

1.2.2 Machine Tool Programming

Once the toolpath has been generated by the slicer program, a series of instructions needs to be constructed so that the machine tool, i.e., the 3D printer, can tell the motors where to move, what trajectory to follow and how fast to move. The industry standard RS-274D(ISO 6893) [6], also known as G-Code, is the most commonly used numerical control programming language to control automated machine tools. Although it is called "G-Code", "G" is only one of many types of commands in this language. Regarding to 3D printing, G-Code consists of G- and M-commands. G-commands tell the control the kind of movement that is wanted. For example, by specifying the endpoints of the move and a velocity, a simple linear movement command can be made starting with "G1". M-commands often call for machine functions or auxiliary actions during the printing process. For instance, "M104" in Marlin firmware sets the hotend temperature.

A 3D printer interprets G-code line-by-line. Each line contains movement instruction, also known as a block. A typical block N_i of G-code for a 3-axis 3D printer is:

$$N_i: \quad G1 \;\; Xx \;\; Yy \;\; Zz \;\; Ee \;\; Ff$$

where x, y, z are the destination coordinates of the move for corresponding axis (if the machine is running in absolute axis mode), e is the destination extrusion volume in total, f specifies the feedrate of the move, which is the maximum movement rate, in mm/minute, of the move between the start and end point. The feedrate applies to the subsequent moves

if f is not specified in the following moves.

1.2.3 Fused Deposition Modeling

Fused deposition modeling (FDM) is one of the latest advances in manufacturing technology. It is one of the most widely used additive manufacturing processes for prototyping and it is likely the first additive manufacturing technology that people are exposed to. For this extrusion-based technology, the material, heated to a semisolid state, is forced out from a nozzle and extruded with a constant cross-section. Then, the material is bonded to solidified material that was previously extruded. Like other AM technologies, the FDM process employs CAD models generated by CAD software. After slicing the tessellated model that is exported from the CAD model, the toolpath is generated from the slicing information [7]. The part is built by depositing semisolid state material in the generated toolpath layer by layer. The materials used in this technology are usually thermoplastic polymers in a filament form. Figure 1.6 illustrates a typical FDM printer. In this system, the printing axis is defined as the z axis. The platform moves inthe z direction driven by lead screws and stepper motors, and the extrusion head moves in the x-y plane driven by belts. The filament is fed and melted in the extrusion head before going through the nozzle and being deposited.

FDM is very easy to use and can print almost any shape that is designed in a CAD software, including those that would be extremely hard to machine. In addition, FDM has a fairly good accuracy considering the price and usability.

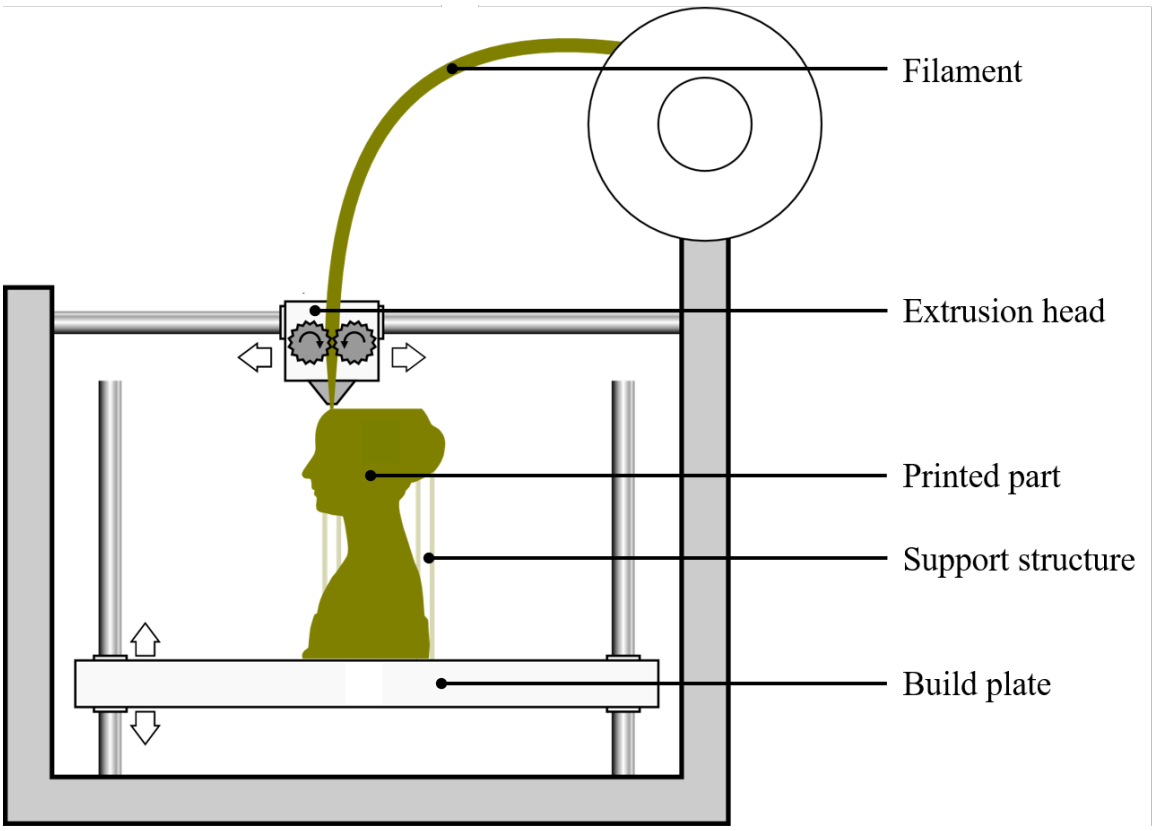


Figure 1.6: A demonstration of FDM printer [8].

FDM Printer

There are three different types of FDM 3D printer in terms of the coordinate system, i.e. Cartesian, Polar, Delta. In this section, these types will be introduced as follows:

- Cartesian Style

  Cartesian FDM is the most common design found in the consumer 3D printer market. As the name implies, the print zone is based on the Cartesian coordinate system that uniquely specifies each point in the zone by x, y and z coordinates that can be taken as the distance between the point and the plane defined by the other two axes. Figure 1.6 demonstrates a design of a Cartesian FDM printer. Usually, this type of Cartesian FDM machine has a build plate that can move up and down in the z axis and a print head moves on both the x and y axes simultaneously and independently. Both the x and y axis have a linear rail that allows the print head to move along and

are driven by 2 belts that are powered by 2 motors with GT2 Gear. On the other side of each axis, there is a bearing that allows the belt to move freely. The z axis is simpler than the x and y axes. It usually only contains 2 sets of leadscrews and linear rails fixed at one end. The other type of Cartesian FDM printer has a build plate that can move back and forth in the y axis and a print head moves on both the x and z axes. The mechanisms are similar to the previous style. The printers of this style have a floating x axis which is a gantry that moves left and right in the x axis. The gantry moves up and down in the z axis with two motors powering two leadscrews simultaneously.

- Delta Style

  Delta FDM printers consist of three motors driving one side of the arms moving along the z axis. Figure 1.7 illustrates a delta style FDM printer. The three arms from each side all connect the extruder head in the middle such that the extruder is suspended by the three arms in a triangular configuration. Even though the configuration is different from that of Cartesian printers, it still controls its movement according to the Cartesian coordinate system. Each position in Cartesian coordinates (x, y, z) can be uniquely transformed to the three axis coordinates in a delta configuration (J_1, J_2, J_3). This can be done by using the inverse and forward pose kinematics solutions [9]. When one arm moves up and the other two move down simultaneously, the extruder moves towards the ip-moving arm. When one arm moves down and the other two move up, the extruder moves towards the center of the up-moving arms. The build plate is stationary at the bottom of the printer. Due to their simplicity in structure, with only three moving axes, Delta style printers can reach higher speeds compared to the Cartesian ones. However, since the moving arms take space vertically, the height of the printer is much greater than the actual print zone.

Figure 1.7: A demonstration of Delta style FDM printer [10].

- Polar Style

  The Polar 3D printer is rarer compared to the other two styles. Figure 1.8 demonstrates an example of a Polar style printer. Instead of using the Cartesian coordinate system, Polar printers use a polar coordinate system in which each point in the space is determined by a distance from the center of the build plate r, an angle from a reference direction ϕ and the height z. This kind of printer usually can achieve a higher build volume within a smaller space without having a framework to move around.

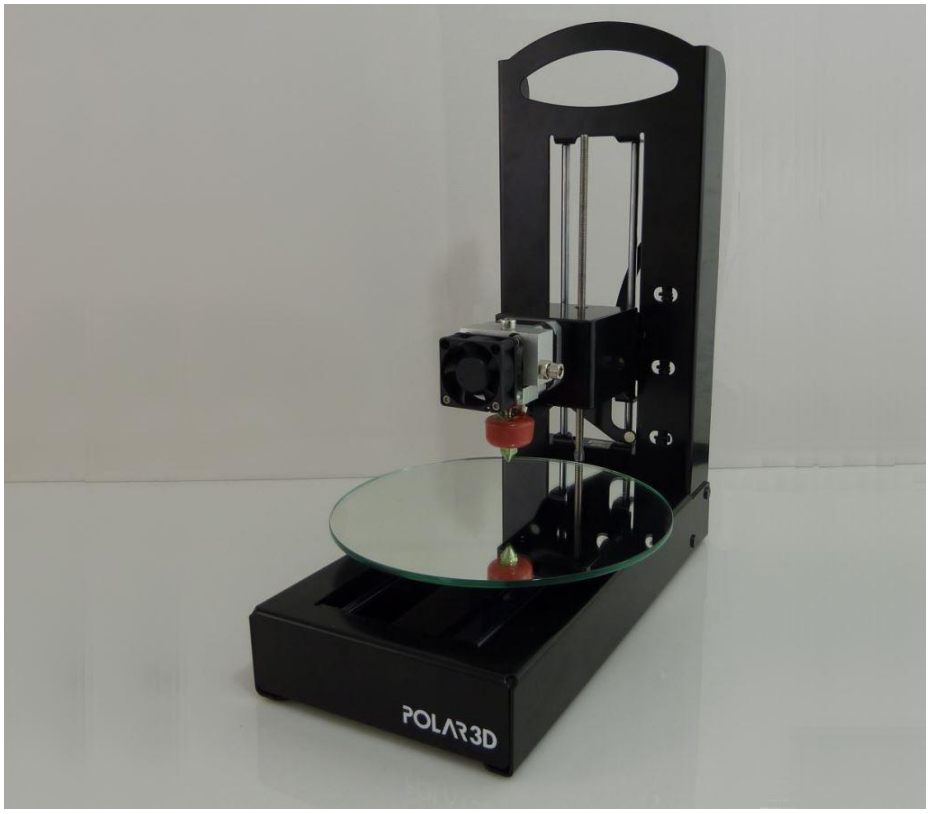
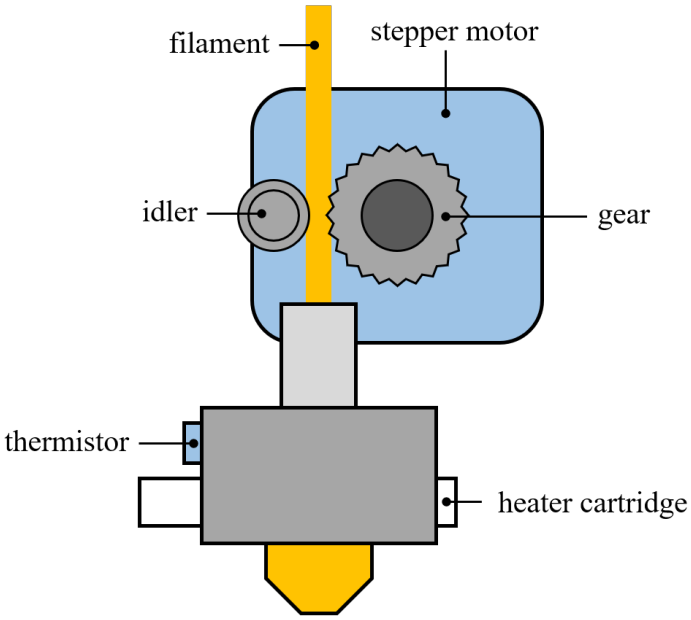
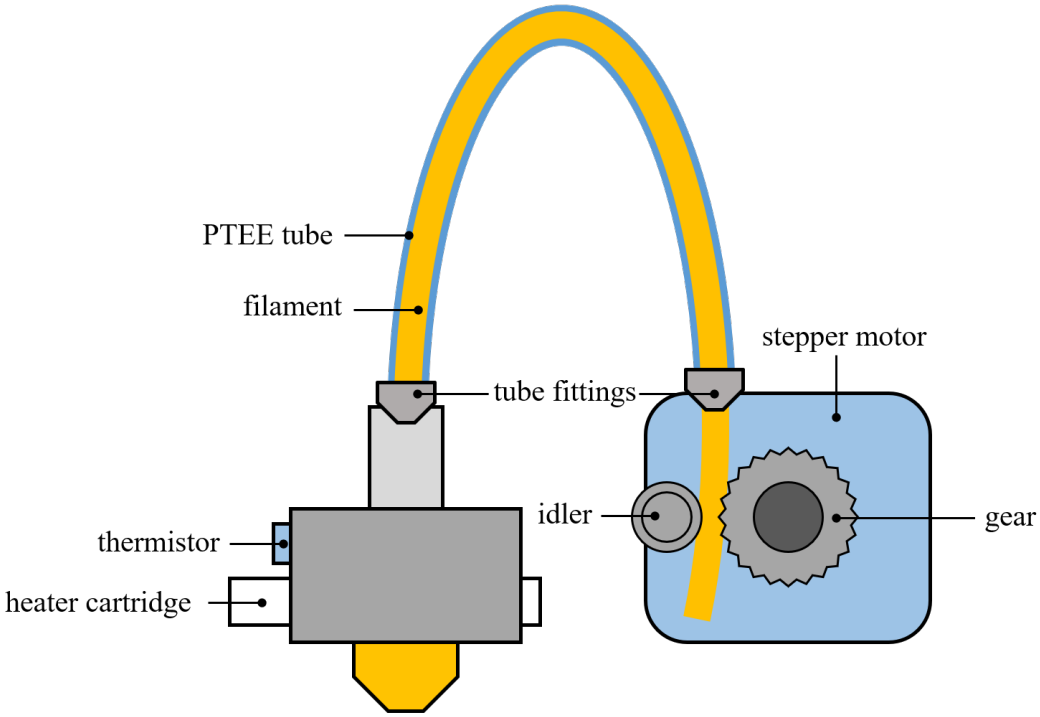


Figure 1.8: A demonstration of Polar style FDM printer [11].

The extruder is an essential component in FDM printers. It consists of two parts: the cold end and the hot end. The cold end refers to the part where the filament is fed and passed along into the hot end. A typical implementation of the cold end consists of a stepper motor, a toothed gearing, and a spring-loaded idler and some tubing to guide the filament. Some improved implementations use two geared hobbed gears instead of an idler to increase the grip to deliver the filament. There are two extruder configurations, i.e. direct drive extruders and Bowden extruders. Figure 1.9 demonstrates these two configurations. As shown in figure 1.9a, in a direct drive configuration, the stepper motor is directly on the top of the hot end. This setup minimizes the distance from the stepper motor to the hot end, resulting in more reliable 3D printing for flexible filaments. The biggest benefit of this setup is better control over the filament retraction with a faster response time.



(a) Before STL model rebuild



(b) After STL model rebuild

Figure 1.9: STL model rebuild.

Another setup, the Bowden setup, is shown in figure 1.9b. The cold end is separated from rather than being mounted on top of the hot end. In this configuration, the stepper motor, which is the heaviest component in the cold end, is moved away from the moving print head; thus the printer will have less of an overshooting problem when the print head is changing direction.

The hot end is the part where the filament transits from solid to semisolid. In a hot end with Bowden configuration, the filament is firstly driven through a tube (usually a PTFE tube). While with direct drive configuration, the filament is usually directly fed into the hot end. Then, the filament is fed into a heat break through a heat sink, which is often a threaded low thermal conductive metal tube (e.g. stainless steel, titanium, etc.). The heat break is to insulate the heat from the heat block to prevent the filament from melting before going to the head block and plugging up the tube, which is known as heat creep. Finally the filament is passing through the heat block. The heat block, often made of aluminum, consists of a heater cartridge, a thermistor and a nozzle. The heater cartridge heats the heat block and reaches the temperature that melts the filament. This temperature is accurately controlled by a PID closed-loop system. Figure 1.10 shows the block diagram of an extruder temperature PID control system. The feedback in the control loop is provided by the thermistor. The nozzle, usually made of brass, has a chamber that tapers to the nozzle's tip. The diameter of the nozzle tip is often 0.4mm for most desktop 3D printers.

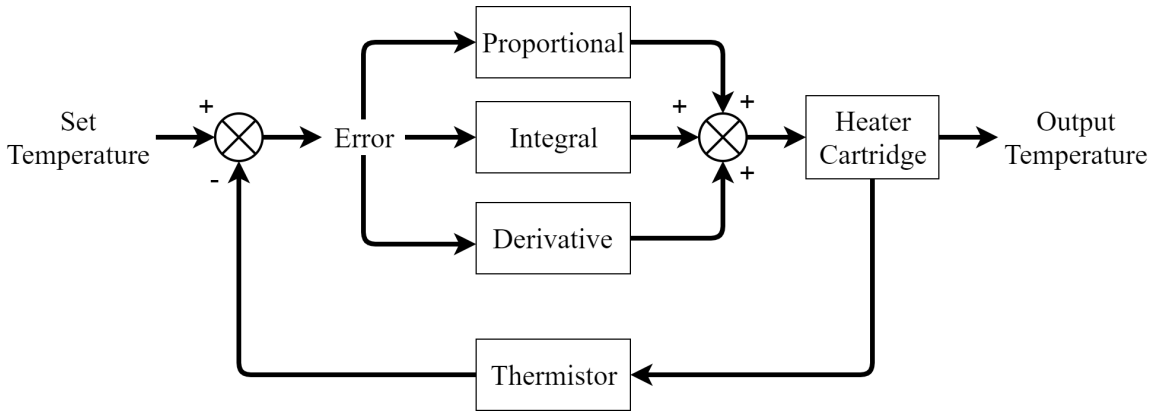


Figure 1.10: A block diagram of a extruder temperature PID control system.

Htin Lin Oo et al. implemented a temperature control system for a MakerBot 3D printer [12]. This system controls the nozzle and build plate temperature using a PID controller. The system was able to achieve the settling time of about 71 seconds with overshoot of 6.5%.

FDM Material

There are various polymers can be used for FDM technology. Among those materials, ABS and PLA are the most common thermoplastics that desktop 3D printers use. Much research has been done for the mechanical properties of the materials used for FDM in different applications. R Melnikova et al. studied the properties of textile-based structure with different polymer materials [13]. It is found that ABS is often too brittle for the textile structure, while soft PLA is proven to be able to construct some fine structures. Sung-Hoon Ahn et al. measured the material properties of ABS [14]. This research performed tensile tests upon ABS with different process parameters. It is concluded that a negative air gap can increase both strength and stiffness, and the stress concentrations occur at radius corners of the tensile specimen due to the discontinuities between printing paths. Sung Hoon Ahn et al. also proposed a tensile failure model of FDM parts [15]. This model can predict failure load as a function of raster angle based on classical lamination theory and Tsai-Wu failure criterion. Ludmila Novakova-Marcincinova and Ivan Kuric summarized some basic and advanced materials used for FDM [16]. This research categorizes the materials into photo-curing, cutting and glueing/joining, melting and joining/binding. Jaroslaw Kotlinski measured the mechanical properties of commercial materials used in different types of additive manufacturing technologies [17]. In this research, various materials (e.g. ABS 2000, ABS plus, ABS 10, PC-ABS, etc.) are tested. The range of the tensile strength is from 36 to 71.64 MPa. N. Mohan et al. reviewed the research carried out for the process parameters optimization for the FDM [18]. It is concluded that there is much research done for improving the mechanical properties of new FDM materials. However, there is still

a need to develop mathematical models that handle factors that can affect the mechanical properties of the part, such as environmental or noise factors.

Limitations of FDM

Even though the most popular choice for 3D printing method today, it still has a number of limitations listed below:

- Materials

  One of the major issues of FDM process is the restricted range of material choices. The parts printed by FDM using these materials cannot meet the demands for more robust parts. These materials are only used for prototyping and demonstration rather than as functional parts.

- Precision

  Another issue of FDM process is the precision of the manufactured parts. Compared to stereolithography (SLA) technology with a layer resolution of around 25 microns, FDM provides poorly detailed objects with thinnest layer thickness of around 100 microns. As a result, the objects printed by FDM technology suffer more seriously from stair-step effect which will be introduce in detail in later sections.

- Final Object Quality The parts' finish is affected when they are removed from the 3D printer. The finish is also suffering from the support structure removal. Often, the bottom surface quality usually suffers the most from the post processes.

- Fixed Building Direction

  The most commercial FDM printers are three-axis machines, namely, the layers are stacking in the z direction. This approach results in a stair-step effect, a lack of strength and the need of support structures. On the other hand, multi-axis FDM systems require innovations in both software and hardware. For the software, the multi-axis trajectory planning needs to be developed to automate computer aided

manufacturing (CAM) processes, which includes innovative slicing procedures and toolpath generation processes. This also introduces miscellaneous optimization problems to achieve required mechanical properties. For the hardware, the conventional extruder head design is not suitable for collision avoidance [1].

1.3 Slicing Algorithms

Once the tessellated model is formatted into STL files, it is ready to be sliced into a stack of layers. This process is effectuated by cutting the 3D model using a series of planes with constant heights, succeeded by transforming these layers to the movements that the 3D printer extruder needs to follow. As a result of stacking flat layers, the printed part will suffer from the stair-step effect [19]. Thinner layers can reduce the geometrical inaccuracy, but this also leads to longer print times. Adaptive slicing [20] was first developed to reduce the stair-step effect without elevating the printing time too much. While adaptive slicing can create surfaces within a specified tolerance, the stair-step effect is not eliminated. To solve this issue, curved layer slicing is then proposed by many researchers. Different methods to offset the top surfaces are developed and evaluated.

1.3.1 Uniform Slicing

The uniform slicing process separates the CAD model into uniform thickness layers for printing [5]. The process starts from the bottom of the model going upward. The slicer cuts the model into XY planes. The XY planes are separated by a uniform layer thickness.

To extract the profile of each layer, every line formed by a triangle intersecting the cutting plane is recorded. The process of searching the intersecting triangles can be optimized by sorting the vertices of the triangles in order of the corresponding Z values. Therefore, the lowest Z vertex and the highest Z vertex can be determined for each triangle. The intersections are only calculated on the triangles whose cutting plane is between the lowest Z vertex and the highest Z vertex. Different situations of the intersection of each triangle

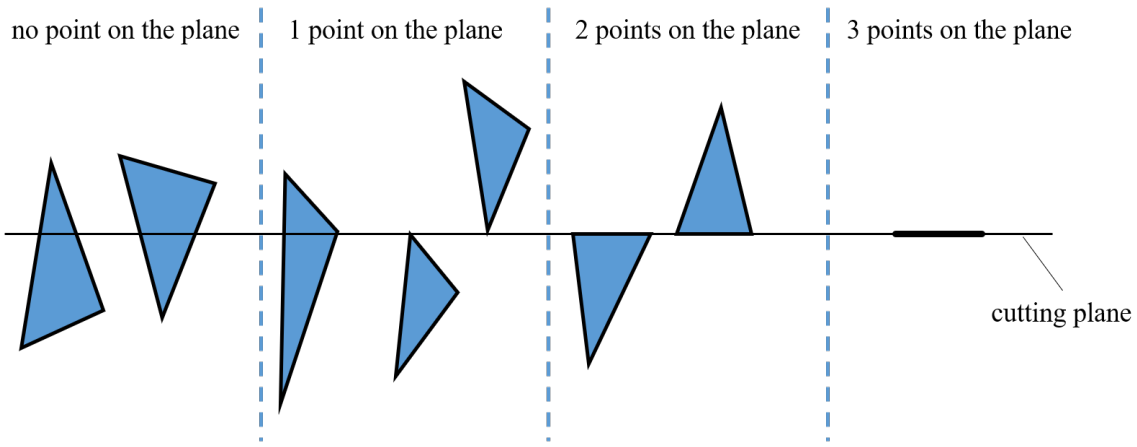


Figure 1.11: Demonstration of different slicing conditions.

with the cutting plane are classified as shown in figure 1.11. For each individual triangle facet:

1. No vertex lies on the cutting plane. In this case, the intersection segment is calculated. The calculation is demonstrated in this section.

2. A single vertex lies on the cutting plane. There is no intersection line in the triangle in this case.

3. Two vertices lie on the cutting plane. The edge corresponding to the two vertices is the intersection that contributes to the profile.

4. Three vertices lie on the cutting plane. The whole triangle is on the plane. The edges that are not shared by two triangles contribute the profile.

Classical line-place intersection methods can be applied to calculate the intersections for the first case. Figure 1.12 shows a general scenario of finding the intersection of the cutting plane and one triangle edge. The two vertices of the edges P_1 and P_2 are defined using Cartesian coordinates in (x, y, z). The cutting plane is in (x, y) with certain z value z_p. The intersection (x_i, y_i, z_p) can be calculated as:

$$x_{i1} = \frac{(z_p - z_1) \cdot (x_2 - x_1)}{(z_2 - z_1)} + x_1 \tag{1.2}$$

$$y_{i1} = \frac{(z_p - z_1) \cdot (y_2 - y_1)}{(z_2 - z_1)} + y_1 \tag{1.3}$$

(x_{i2}, y_{i2}) can be determined similarly. Therefore, the intersecting lines of every triangle and the cutting plane can be determined.

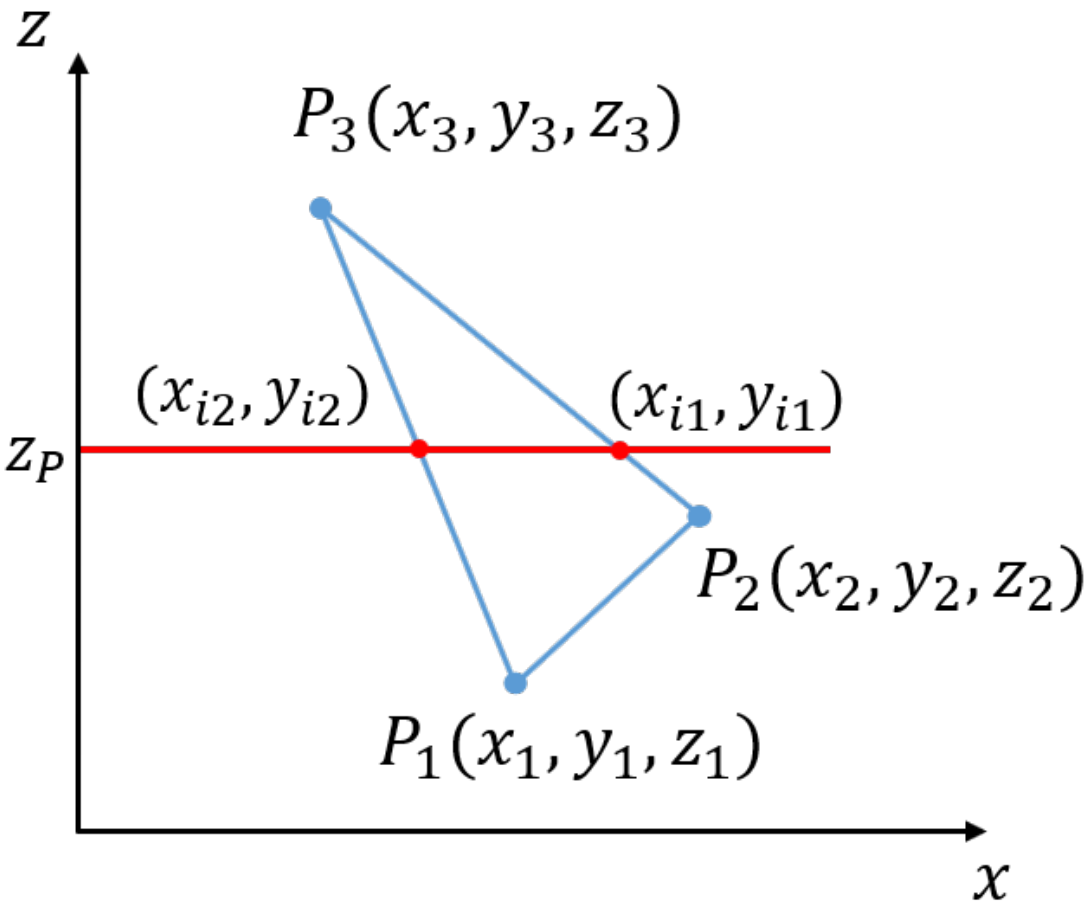


Figure 1.12: Demonstration of intersection calculation.

As the triangles contained in a STL file can be randomly distributed, checking every triangle with each cutting plane can be computationally inefficient. Therefore, a pre-process can be built for better efficiency. One way to speed up the search for triangles that are cut with the cutting plane is to sort the triangle vertices in the order of z-value. After sorting the z-value for each triangle. A simple check that can be applied for each cutting plane would be to check the z-values of the vertices of each triangle. If the z-value of the cutting plane

is between the minimum and the maximum z-value of the triangle, then that triangle should intersect the plane. The intersection lines can be determined using the method described above. The process is shown in figure 1.13.

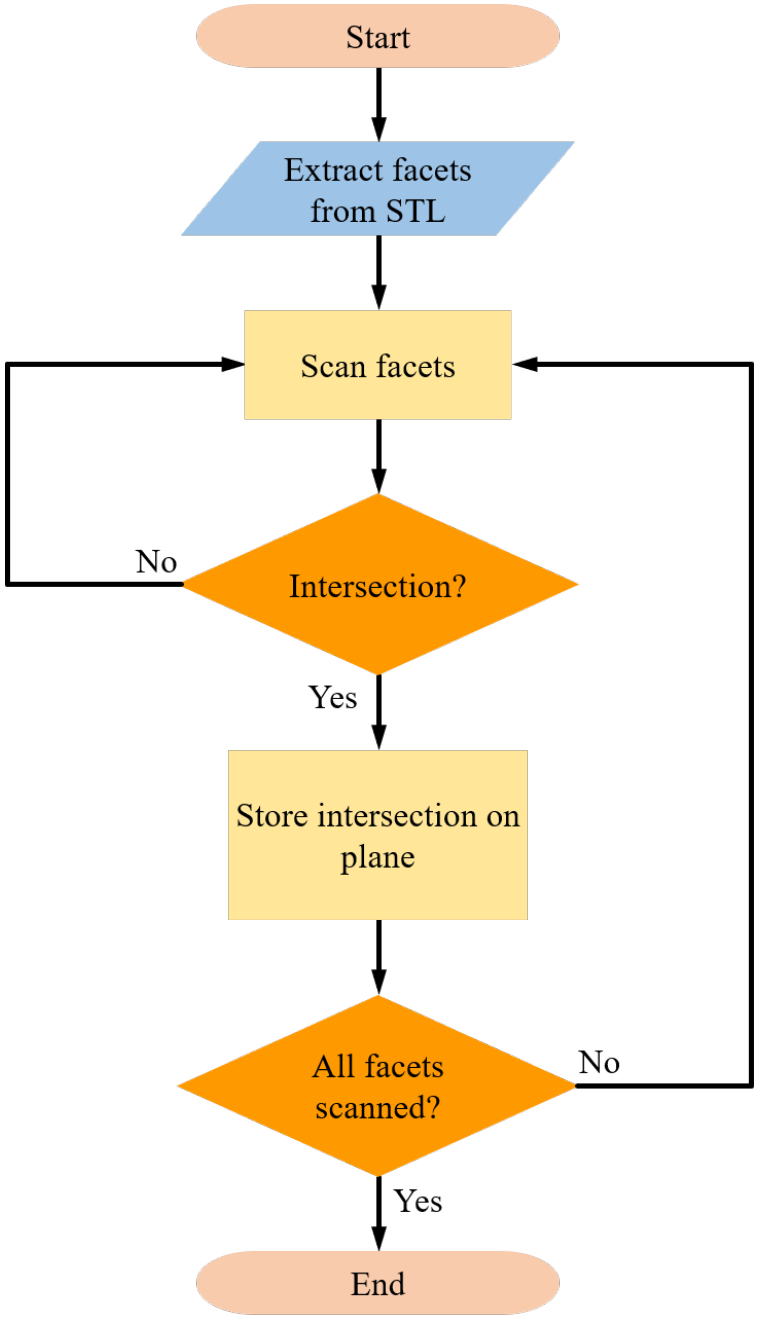


Figure 1.13: Algorithm for determining intersections from STL data.

Once all intersection line segments have been calculated, they need to be connected to

form polygons which represent the contours of the objects. Based on the process present in figure 1.14. The idea of this algorithm is to find the closest line segment from the current line segment. A tolerance may be introduced to determine if a point is close enough to be considered as a connecting point because the closest points may not have the identical coordinates that determined from the intersection calculation.

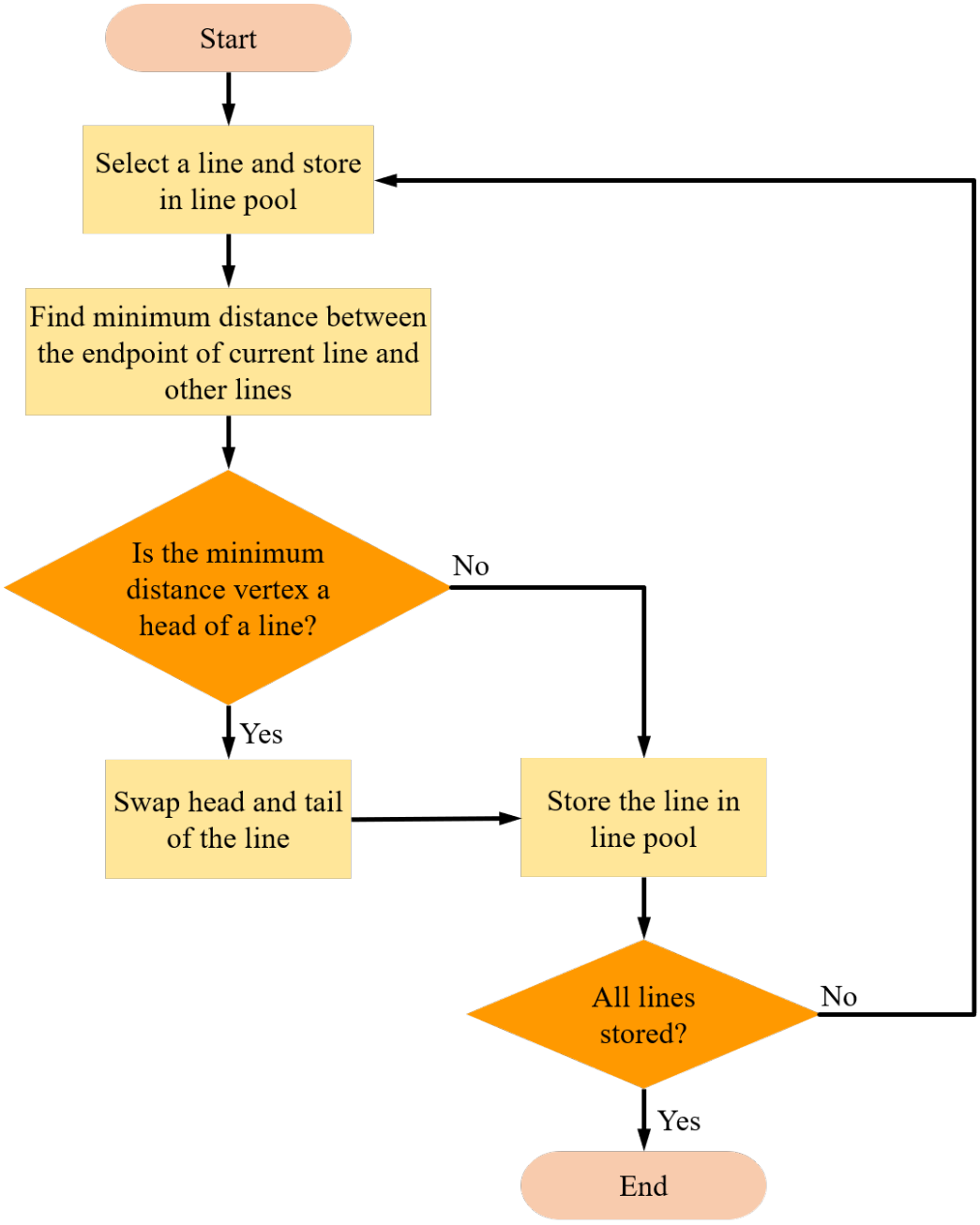
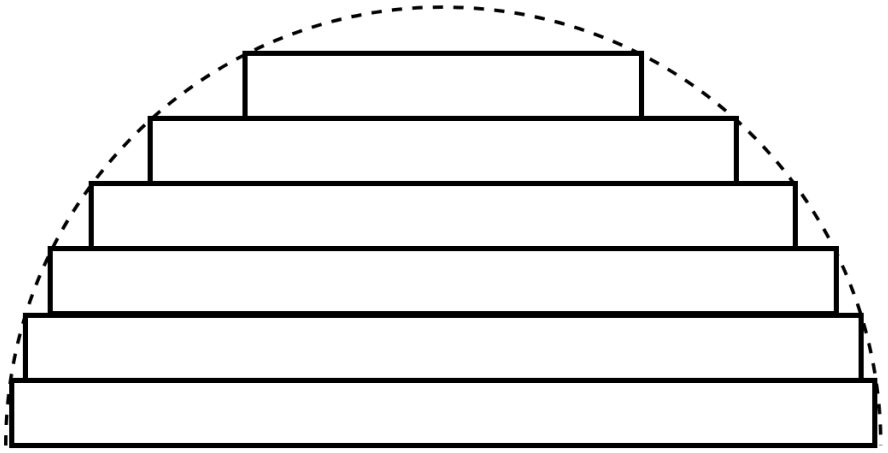
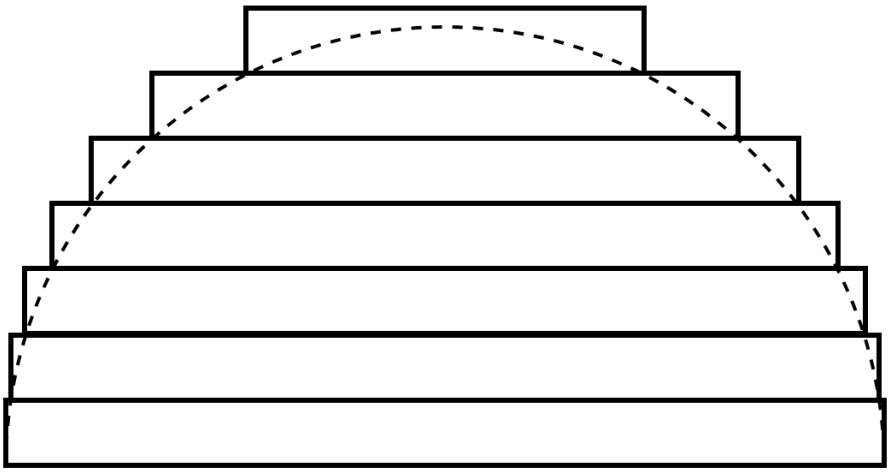


Figure 1.14: Algorithm for connecting intersections.

1.3.2 Stair-Step Effect



(a) Inside stepped edge



(b) Outside stepped edge

Figure 1.15: Two types of stepped edges.

The stair-step effect is inherent to the uniform slicing process and occurs due to the existence of the stepped edges. There are two types of stairs, outside and inside stepped edges, as shown in Figure 1.15. In this representation, the contour of the layer edges is considered squared. The presence of the stair-step effect is one of the major concern for the quality of the prototype. Decreasing the layer thickness could improve the surface finish at the cost

of a longer build time [1].

Many researches have studied modeling of the stair-step effect. Evren Yasa et al. proposed a model for the stair-step effect in direct metal laser sintering [21]. This research models and predicts the stair-step effect using a numerical approach. The contour of the layer edges is considered as circle with a diameter of a layer thickness. C.J. Luis Pérez et al. characterized the roughness in additive manufacturing [22]. The roughness average (Ra) is calculated for 2 different models, i.e. squared edge and rounded edge, in this research. Based on the Ra, the constant layer thickness can be obtained to achieve the required tolerances. Daekeon Ahn et al. attempted to represent surface roughness in FDM objects [23]. In this research, a theoretical model to represent surface roughness distribution under different surface angle is proposed and verified. The cross-section of the deposited filament is considered to be elliptic. The filaments in successive layers are stacking and overlapping. This model is validated by comparing the measured data and predicted data. Sang-in Park and David W. Rosen developed a numerical model for evaluating the impact of geometric errors on mechanical properties using voxel modeling approach [24]. This research considers the stair step effect as a factor that affects the mechanical property. A voxel based finite element model is proposed and used in this approach to simulate tensile tests. A. Boschetto, V. Giordano and F. Veniali proposed a geometrical description of profile roughness [25]. The roughness average (Ra) in this geometrical model can be numerically calculated from layer thickness and stratification angle.

Some researchers have tried to eliminate the stair step effect by applying secondary finishing operations, which is also known as post processing. Pulak M Pandey et al. proposed a CNC milling method to improve surface roughness [26]. This approach can be time-consuming, as it needs machine setups and operations. Some complex objects can be impossible to machine due to inaccessible features. Robert E. Williams and Vicki L. Melton proposed an abrasive flow machining (AFM) approach to finish additive manufactured objects [27]. Similarly, Kah Fai Leong et al. proposed an abrasive jet deburring

method to finish stereolithography apparatus objects [28]. These approaches attempted to find the best machine setup and process parameters to achieve better surface finish with acceptable machining time. Alberto Boschetto and Luana Bottini proposed approach using a barrel finishing (BF) to improve the surface roughness for FDM objects [29].

1.3.3 Adaptive Slicing

To achieve accurate surface geometry without any secondary process, much research has focused on finding optimal layer thickness for each layer to slice a model. A. Dolenc and I. Makela introduced cusp height tolerance concept and attempted to restrict the stair step effect to a user-defined cusp tolerance [30]. Figure 1.16 demonstrates the idea of adaptive slicing. The layer thickness is determined by a user defined geometrical tolerance. The error between the CAD model and the deposited part is defined in terms of a cusp height tolerance. As shown in figure 1.16, the build edges are considered rectangular, and the layer thickness, t, is determined by a pre-defined maximum allowable cusp height. The desired layer thickness can be calculated by

$$t_d = \min\left\{L_{max}, \frac{C_{max}}{N_z}\right\} \tag{1.4}$$

where C_{max} is the maximum allowable cusp height, N_z is the z component of the normal vector of the surface, and L_{max} is the maximum layer thickness that the AM machine can produce. And the slicing layer thickness is given by

$$t = \max\{L_{min}, t_d\} \tag{1.5}$$

where L_{min} is the minimum layer thickness available.

The adaptive slicing procedure has been demonstrated by many research applications towards part improvement. Sabourin, Houser and Bøhn proposed a stepwise uniform refinement adaptive slicing method [31]. First, the CAD model is sliced with maximum

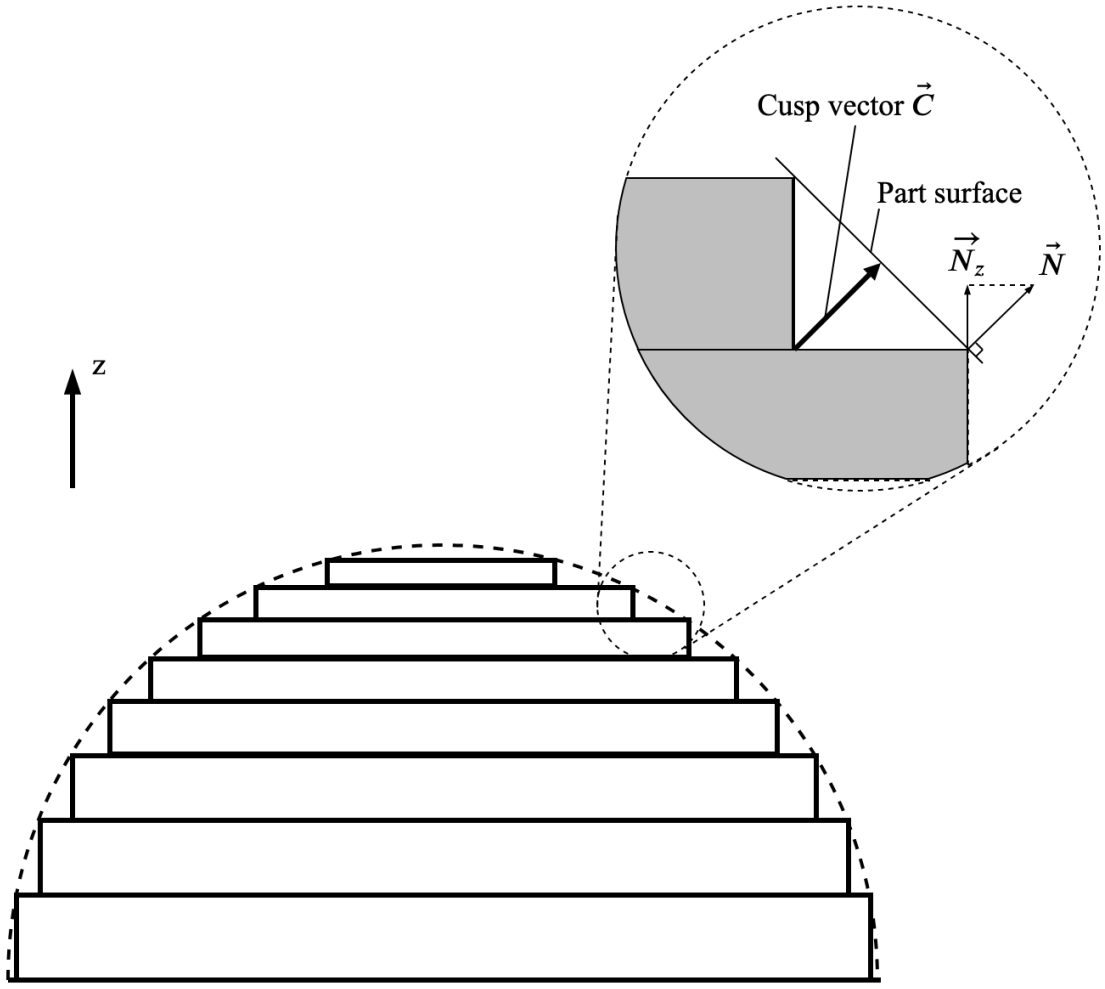


Figure 1.16: Adaptive slicing and cusp height.

available layer thickness using uniform slicing algorithm. Then, each layer is re-sliced into sub-layers to achieve the required cusp tolerance. Tyberg and Bøhn introduced a local adaptive slicing algorithm [32]. This research dynamically slices the model for each local feature. This approach increases the print efficiency significantly by avoiding the slices that do not improve the surface quality. Mani, Kulkarni and Dutta proposed a region-based adaptive slicing algorithm [33]. The idea is similar to [32], which treats different regions in the part with different cusp tolerances. This improves the overall efficiency in another way that doesn't sacrifice the surface quality. Pandey, Reddy and Dhande attempted to

adaptively slice the model based on the parabolic layer edge profile instead of square [26]. This method calculates the layer thickness in real-time based on the previous layer edge profile and cusp tolerance. Hope, Jacobs and Roth introduced an adaptive direct slicing system based on sloping surfaces criterion [34]. This approach describes the stepped edge profiles using B-spline surface and evaluate the surface error by measuring the distance between the B-spline surface and the cutting vector. Ma, But and He proposed an innovative approach to slice NURBS-based models using adaptive slicing and selective hatching strategy [35]. In this research, the peak features are identified and kept during the adaptive slicing procedure. The selective hatching module then computes the hatching area to separate the internal region and the skin region and apply different layer thickness on these two regions. Zhang and Liou developed an adaptive slicing method for multi-axis additive manufacturing models [36]. This approach optimizes the deposition direction to minimize the support structure and builds parts in a 5-axis hybrid system. Adaptive slicing is applied to every deposition direction to maximize the efficiency. Hayasi and Asiabanpour proposed an adaptive slicing method [37]. Instead of the maximum available thickness, this algorithm starts with the minimum available thickness such that any concave or convex corner on the object profile can be represented as accurately as possible after slicing. Then the layer thicknesses are determined based on area deviation and triangle area tolerances of the contour on top and side views. Wasserfall, Hendrich and Zhang developed an adaptive slicing system based on the volumetric tolerance rather than the 2D cusp tolerance [38]. This research categorizes the surface deviation into stair-step effect and surface roughness caused by surface slope and layer thickness. The final layer thicknesses are determined to keep the total volumetric deviation within the desired tolerance.

1.3.4 Curved Layer Slicing

Other than adaptive slicing procedures, much research has focused on curved layer slicing to address some of the major limitations in flat layer slicing, e.g. stair-step effect and

discontinuous toolpath on the top surface. Klosterman et al. proposed a curved layer LOM process to manufacture curved layer objects, especially thin curved-shell components [39]. The z value of each point on the curved layer is interpolated from a "height grid". The shape of each new layer is determined by an open-loop method, which offsets a point on the grid with adjacent triangles along with the normal vector of each triangle by the distance of a layer thickness. Then, it fits a surface tangent in the desired four offset triangles with a third degree polynomial. Chakraborty, Reddy and Choudhury proposed a toolpath generation algorithm for a curved layer fused deposition modeling (CLFDM) process [40]. The geometry of the filament path is formulated and simulated in this research. Huang and Singamneni integrated adaptive slicing and curved layer slicing based on three-plane intersection method for curved layer offsetting [41]. This method can handle simple shapes to achieve adaptive curved slicing.

Some other research has attempted to model and implement the CLFDM for various applications. Diegel et al. discussed the possibility of applying CLFDM to plastic components with conductive electronic tracks [42]. The CLFDM technology has the potential to build such plastic parts without printed circuit boards and wiring. A proof-of-concept machine was built in this research to validate the hypothesis. Allen and Trask implemented the curved layer fused filament fabrication method on a delta style 3D printer [43]. A parameterized skin surface is manufactured as an example part in this research. The toolpath is generated by calculating the static z value on the surface with known x and y coordinates. The surface finish is significantly improved compared to that of a flat layer sliced part. Lim et al. implemented curved-layer additive manufacturing for a large-scale construction process [44]. In this research, the toolpath is generated in a plugin of Rhinoceros and converted to G-code. An example is then printed and evaluated using the 3D concrete printing system.

The key steps in developing a curved layer slicing algorithm are to collect the vertices and facets on the top surface of the part. After grouping the point cloud of the top surface,

the facets and vertices are offset along the normal direction by an amount equal to a layer thickness. As mentioned above, there are different techniques to determine the normal directions depending on the application.

1.3.5 Direct Slicing

Although the STL format is widely used in industry, there are other ways of defining 3D models and of generating the slice data. In some specific fields, for instance, tissue engineering that fabricate tissue scaffold structures, the final parts are remarkably impacted by the accuracy of the geometrical representation of the CAD model [45]. The bio-mimetic scaffold structures are designed to replace actual body sections, which requires a more accurate representation of CAD models than STL format. Also, for applications that produce large axis-symmetric or spherical geometries, the STL files are usually larger than the CAD file due to its high redundancy in geometry representation. Hence, generating slice data directly from CAD tools by calculate the intersection for a plane with a model would benefit such applications.

Many researchers have attempted to develop a direct slicing method based on one of the CAD software packages. Those CAD softwares provide slicing packages or support slicing commands in different ways. Chen, Wang and Ye developed a direct slicing method from PowerShape, which is a CAD software for complex part modeling. The models are sliced into layers by writing a macro file, which contains the slicing commands, to the AutoSection, which is a built-in package in PowerShape [46]. Cao and Miyamoto proposed a direct slicing method from AutoCAD solid models [47]. This method sends message written in VBA to AutoCAD to utilize the AutoCAD ActiveX Automation interface, which provides a SLICE command. The sliced planar data are stored in a DXF file.

Some researchers have tried to develop slicing methods independent of any CAD software. Starly, Lau and Sun developed a direct slicing method for STEP based models represented by NURBS surfaces [48]. This method determines the optimal build direction

by minimize the build height; refines the NURBS surfaces by adding more control points without changing the original shape to guarantee the convergence occurs within the refined sub-patch; finds the intersection points by bisection iteration routine, then, categorizes the intersection points into entry and exit. This method is independent of CAD software, as it is based on a standard format (STEP) that is supported by most CAD software. Oropallo, Piegl and Rosen proposed a point based slicing approach [49]. This algorithm first discretely samples the original model and converts it to a point-based representation. Second, the point-based model is sliced into groups of points, which are within a layer thickness in the z direction of the layer height. Then, the layer points are separated into intersection curves, and the boundaries of the curves are fit with B-spline curves. This method bypasses the difficulty of slicing the NURBS model by transforming it to point-cloud model.

Prior work has also been explored in the area of adaptive direct slicing. Zhao and Laperriere implemented an adaptive direct slicing method that integrates adaptive slicing with direct slicing [50]. This method reads the DXF file generated from AutoCAD, slices the model into 2D contours using adaptive slicing to guarantee the desired staircase tolerance, then generate tool paths. Zhou, Xi and Yan developed an adaptive direct slicing method with non-uniform cusp heights based on STEP format[51]. In this method, different quality requirements can be satisfied for various part surfaces. Sasaki et al. developed an adaptive direct slicing method for heterogeneous objects [52]. Two types of heterogeneous objects are considered in this research, functionally gradient material (FGM) and multi-material (MM). The heterogeneous models are first represented by fitting volumetric attribute data by trivariate B-spline functions. Then, adaptive direct slicing technique is applied to the model. Sikder, Barari and Kishawy proposed an adaptive slicing method based on surface integrity [53]. In this method, a surface error is defined to quantify the geometrical error for each layer. The layers are generated one-by-one until the error of each layer is smaller than the desired allowable error. Zheng et al. developed an adaptive direct slicing method based on tilted voxel [54]. This method is used for two-photon polymerization (TPP), which is

a micro-machining approach. The ellipsoid-like voxels are tilted based on the changing of the curvature on the contour lines. Feng et al. proposed a direct slicing method based on T-spline models [55]. As claimed to be the generalization of NURBS, T-spline can add more control points without traversing the control grid. Thus it can be used to design more complicated models. This research applied adaptive slicing algorithm to T-spline models to slice T-spline surfaces with a iteration algorithm.

The major drawback of direct slicing is that the 3D model representation varies from CAD system to CAD system. Even the most commonly used format, e.g. STEP, is only supported by a few CAD softwares. Those slicing methods that rely on a specific CAD system cannot be used for other CAD systems.

1.4 Variable Thickness Layer Slicing

1.4.1 Problem Statement

Although much research has been focused on changing the layer thickness for each layer to achieve better surface quality, the curved layer slicing process can potentially eliminate the stair-step effect. However, not many curved layer additive manufacturing applications have been implemented due to restrictions on geometrical features. The proposed research is to eliminate stair-step effect and to slice models including parts that require variable thicknesses within each layer.

1.4.2 Research Objectives

This dissertation aims to develop and evaluate a slicing algorithm for tessellated models that reduces the geometrical error in the final part that is caused by the stair-step effect. Realization of the path planning and G-Code generation system will provide the capability to convert the STL model to the final part. The slicing algorithm and its evaluation requires development and construction of FDM printing hardware and software, which will be introduced in detail. A successful slicing procedure for base case will handle the models with

several constraints, while the one for general cases will handle less constrained models by deconstructing the model into two types of sub-cases. The slicing procedure will produce parts with less geometrical errors compared to uniform slicing algorithm.

Contributions

The development and evaluation of this system will enable the following functionality that is not present in FDM systems that use uniform slicing or adaptive slicing algorithm:

1. The curved top surface of the model will be produced by a single layer which provide better accuracy in geometric.

2. The procedure has the potential to be applied on 5-axis FDM machines by simply adding two rotational axis angles to each path, which can be easily calculated based on the normal vectors of the facets.

3. The total print time will be reduced by using fewer number of layers compared to adaptive slicing algorithm.

Evaluation Metrics

The measure of success in this realization will be the geometrical accuracy of the final part. The error caused by the FDM machine itself is first evaluated by measuring the flatness of the parts that have flat top surfaces. Three sets of test parts with different slope angle are then printed with both variable thickness layer slicing and uniform layer slicing. The geometrical accuracy of these parts is evaluated by measuring the top surface's geometrical characteristics using a coordinate measuring machine (CMM). Finally, a tensile test is performed to evaluate the tensile strength in the z direction.

Assumptions

This dissertation is made based on a number of assumptions, each of which is listed below:

1. The FDM machine that is used in this dissertation is repeatable and stable.

2. The STL models sliced by the proposed procedure are printable. No STL repairing is needed.

3. The coordinates measurement obtained from the CMM is perfectly accurate.

4. The tensile strength measurement obtained from the tensile test machine is perfectly accurate.

Limitations

A number of limitations will be disclaimed in this the demonstration of the proposed slicing procedure, each of which is listed below:

1. The proposed slicing procedure does not work on the models that need support structures that are placed on or inside the model. In other words, the proposed procedure can be only applied to models that do not need support structure or all support structures are directly touching the build plate.

2. No support or any advanced printing features (i.e. multiple extrusion, travel combing, etc.) are implemented in this research, while there is no physical or theoretical basis for realizing those features. Therefore, this may result in unexpected part surface finish compared to those commercial mature slicing softwares.

3. The bottom surface finish still depends on the build plate surface conditions and support structure surfaces that is in contact with the parts.

4. When the proposed procedure is applied on the 5-axis machines, the rotation of the part or nozzle can cause interference, which needs additional considerations.

CHAPTER 2

EXPERIMENTAL EQUIPMENT AND SOFTWARE

This section describes the experimental setup of the mechanical systems and software for developing and implementing the slicing algorithm proposed in this research. The main components are a computer, a machine monitoring and control system, a 3-axis Cartesian FDM machine and its related hardware and software.

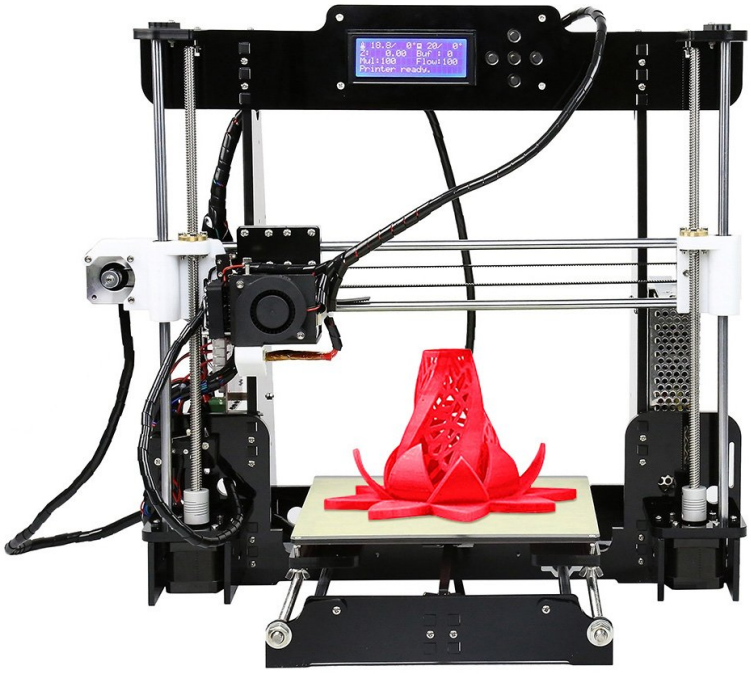


Figure 2.1: Stock Anet A8 3D printer.

2.1 Computer

A computer in this research is used to monitor the status of the FDM printer and to provide a user interface for the printer. It is also used to run the slicer to generate G-code from

3D models. The computer is equipped with an Intel i7-8750H CPU, 16GB of memory and NVIDIA GeForce GTX 1050Ti GPU. The operating system of the computer is Windows 10. The computing power of this computer is sufficient to design parts for experiments in SOLIDWORKS, run the slicers and generate G-codes, monitor and communicate with the printer controller, and analyze the data.

2.2 FDM Printer

A FDM printer, known as the Anet A8, is responsible for printing the desired objects in this research. Anet A8 is an inexpensive alternative of Prusa i3 MK3, which is the benchmark for desktop 3D printer kits. Ainengt Technology Co., Ltd. manufactures 3D printers and sells Anet A8 for approximately $170. A stock Anet A8 is shown in figure 2.1.

The Anet A8 is a Cartesian style FDM machine, in which the extruder moves in the x and z directions while the build plate moves in the y direction. The frame of the Anet A8 is made of Acrylic. It uses linear bearings, belts and threaded rods to build on the x, y and z axes. The Anet A8 printer is sold unassembled. Putting the printer components together and calibrating the printer requires time and patience. Some specifications of the printer are given in table 2.1

Table 2.1: Anet A8 specifications.

Metric		Unit	Value
Build size		mm	220*220*240
Resolution	X	mm	0.05
	Y	mm	0.05
	Z	mm	0.015
Filament diameter		mm	1.75
Speed		mm/s	10-120
Layer thickness		mm	0.1-0.3
Nozzle diameter		mm	0.4

The motherboard of the printer is shown in figure 2.2. This motherboard features 16-

step A4988 stepper motor drivers. The on-board drivers allow tuning the motors through the firmware by adjusting the current. This motherboard is officially supported by the Marlin firmware, which will be described in a later section.

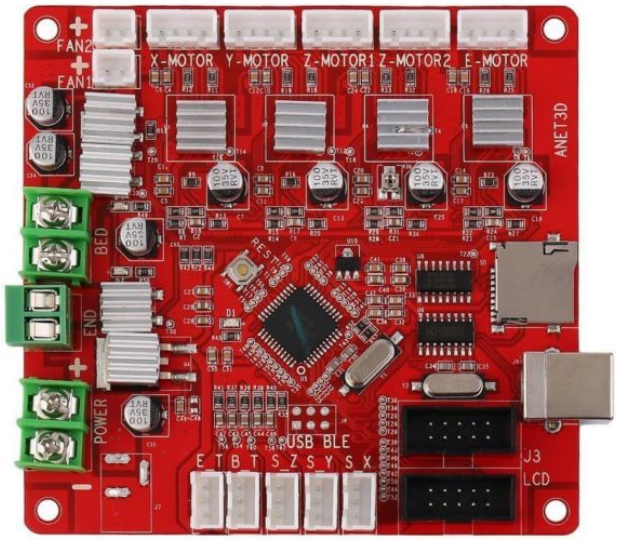


Figure 2.2: The motherboard of the Anet A8 3D printer.

2.3 3D Printer Firmware

A 3D printer firmware provides low-level control for the printer. The firmware, known as Marlin, runs on the motherboard of the printer. Marlin is a completely open-source firmware that controls all of the 3D printer's real-time activities, including the movement in 3 axes, I/O control tasks, etc. The firmware takes G-code and interprets the commands into the movements. Over 150 commands, including G-commands and M-commands, are supported. The heater temperature is also controlled by this firmware with PID closed-loop controller described in section 1.2.3.

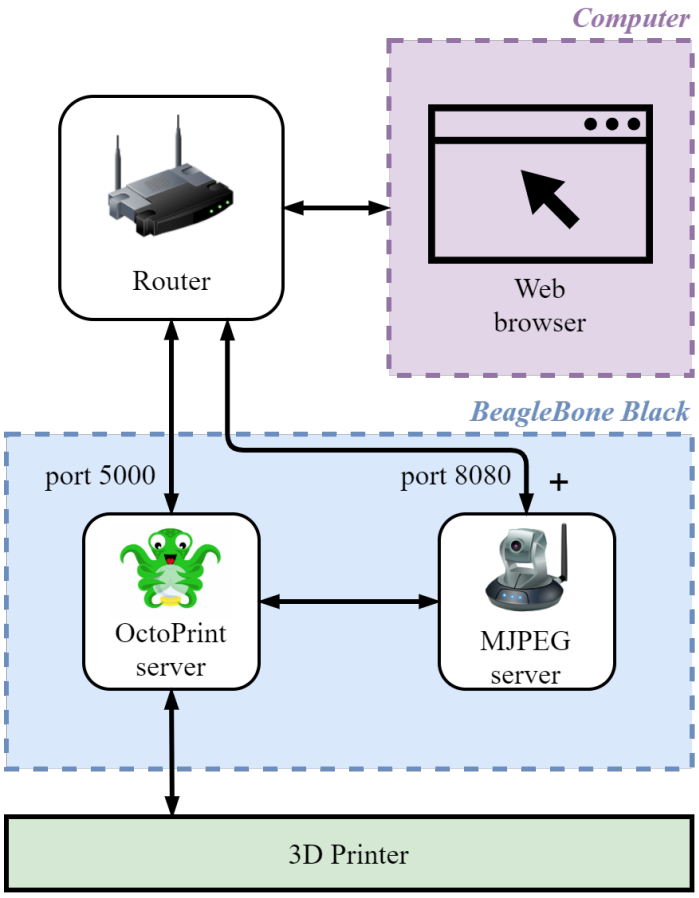


Figure 2.3: Printer monitoring and control system.

2.4 Printer Monitoring and Control System

OctoPrint is an open-source host for 3D printers that provides a web interface for users to control and monitor the 3D printer through a browser. This host is run as a server on an single-board computer, the BeagleBone Black Wireless (BBB), that runs a Debian Linux operating system. The OctoPrint connects the printer through a USB cable and directly sends G-code commands to the printer. The BBB connects the router through its built-in Wi-Fi component; therefore, any computer in the local network can connect to the OctoPrint server through BBB's IP with port 5000. The network structure is represented in figure 2.3. This system provides not only real-time control over the printer while printing, but also live viewing of prints through a webcam. The MJPG streamer server is hosted

on BBB with port 8080. The OctoPrint integrates the video streamer into its interface and provides extra features, e.g. timelapse video creation. This allows fully remote control over the printer from anywhere by setting up the port forward in the router.

2.5 Slicing Software

The slicing software used in this research is Cura, which is a general-purpose open-source software, developed by Ultimaker BV, under LGPLv3 license. It consists of two parts, the front-end, written in Python, which provides the user interface and model visualization, and the back-end written in C++, known as CuraEngine, is the actual slicer that slices the model and generates the G-code. The print settings can be specified in the front-end. The settings are stored to a JSON file that will be sent to CuraEngine. Cura also has a plugin system that allows third-party extensions to be integrated into the front-end, which makes it a user friendly and versatile 3D slicing software. The interface of Cura 4.0.0 is shown in figure 2.4

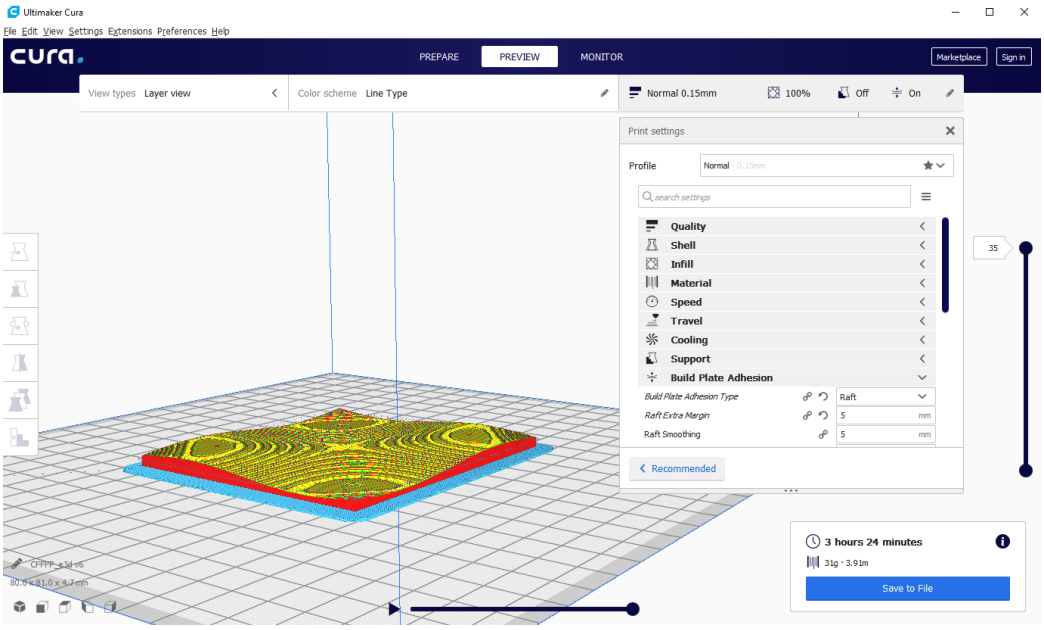


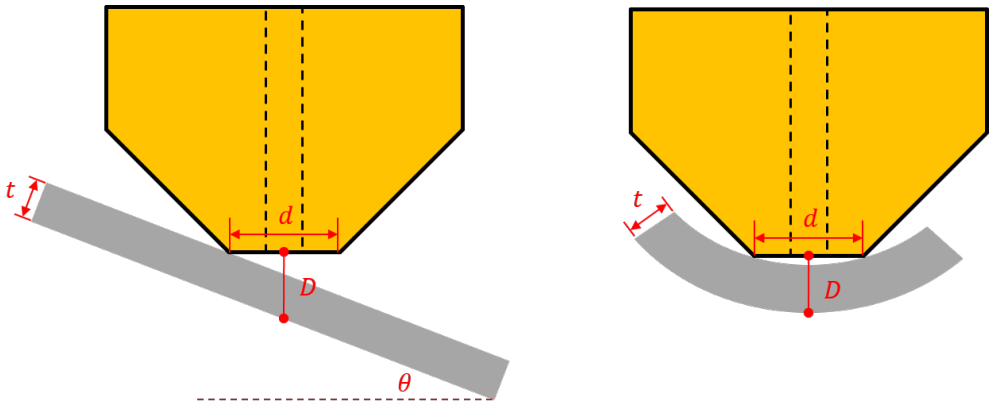
Figure 2.4: The interface of Cura 4.0.0.

2.6 Nozzle Modification

In flat FDM, the nozzle tip contacts with the top surface of the deposited part with a distance from the former layer by the layer thickness t. Alternatively, in CLFDM, the distance between the nozzle tip and the former layer must be larger than the layer thickness to avoid interference between the nozzle and the deposited part. Figure 2.5a demonstrates the geometrical relationship between the nozzle and the deposited surface. The distance between the nozzle tip and the former layer for FDM can be calculated by:

$$D = \frac{t}{\cos\theta} + \frac{d \cdot \tan\theta}{2} \tag{2.1}$$

where t is the layer thickness, d is the nozzle tip diameter and θ is the angle between the nozzle tip surface and the tangent direction of the surface.



(a) Nozzle and the deposited surface (b) Nozzle interference

Figure 2.5: The distance between the nozzle tip and the former deposited layer.

By using a 5-axis machine, this issue can be partially solved by constraining the deposit axis to be normal to the deposit surface. However, any concave surface can still cause interference between the nozzle tip and the former layer, which is shown in figure 2.5b. This may lead to ineffective bonds between layers and reduce the strength of the part. In addition, calculating the curvature at each point and, therefore, the desired distance between the nozzle tip and the former layer can result in an extreme computational burden.

To address this problem, a modified nozzle will be used. The sharp corners around the nozzle tip will be ground to produce fillets tangential to the original edges. The solid lines in figure 2.6 show the original shape of the nozzle, and the dashed line represents the modified shape. The nozzle is ground on a lathe to make this modification. The threaded tail of the nozzle is mounted in a collet. A piece of sandpaper is used to grind the tip edge of the nozzle. Lastly, the inner surface of the center hole in the nozzle is deburred by a 0.4mm drill bit.

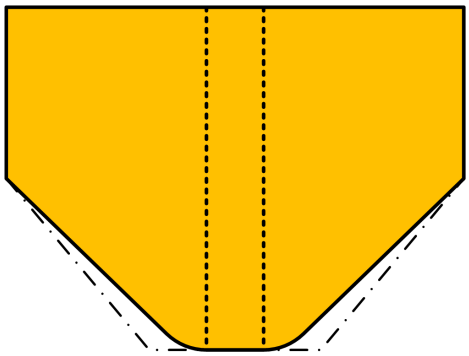


Figure 2.6: Extruder nozzle modification.

CHAPTER 3

SLICING ALGORITHM FOR BASE CASE

This chapter introduces the slicing algorithm for the base case. In this research, a base case model is constrained under 3 assumptions: 1. No support structure; 2. The model can be partitioned into Uniform Thickness Region (UTR) and Variable Thickness Region (VTR) (to be explained later in this chapter) by a dividing planar, and the entire part can be manufactured with the same number of layers; 3. The layers in a particular section must all be exclusively UTR or VTR. The model used as an example in this research is a hemisphere. The STL model of a hemisphere is shown in figure 3.1.

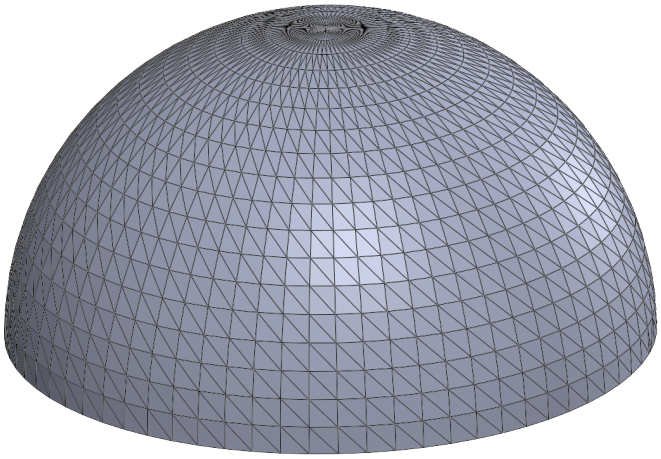


Figure 3.1: Tessellated hemisphere model.

The slicing procedure consists of three major steps: first, the STL model is read and preprocessed; second, the UTR and VTR are determined; finally, a series of layers is generated in both regions. This process is shown in figure 3.2

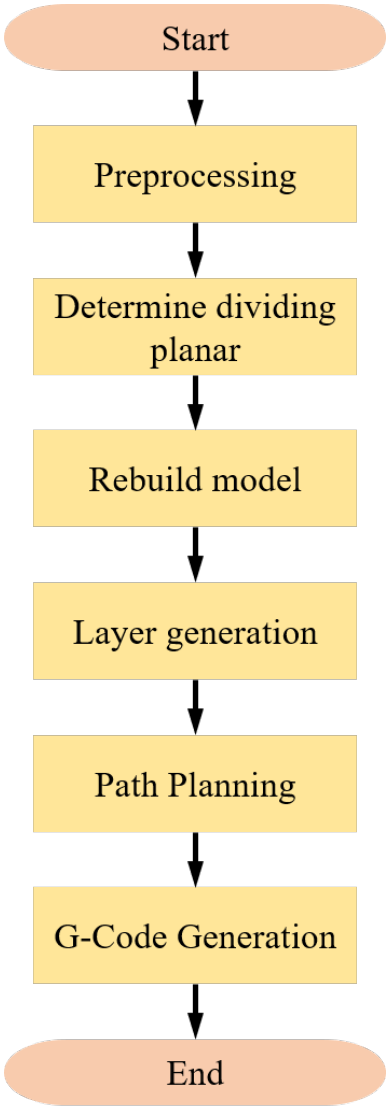
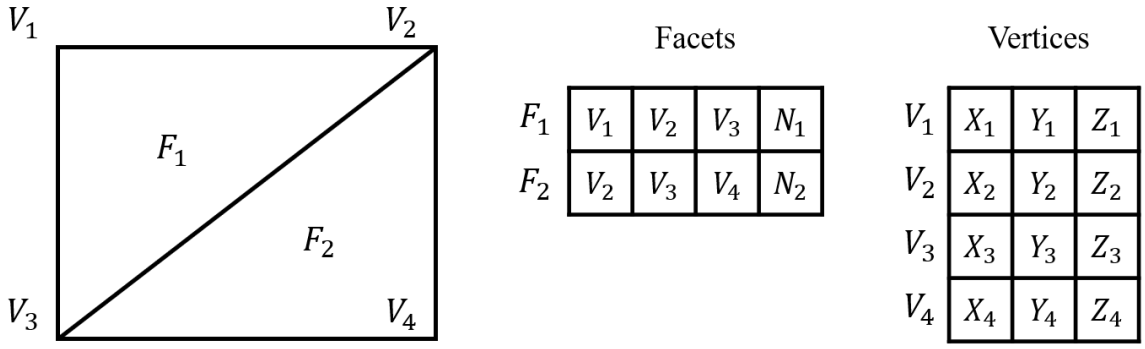


Figure 3.2: Slicing procedure for base case.

3.1 STL Model and Preprocessing

The very first step of slicing is importing geometry from an STL file into a set of facets and vertices. As introduced in table 1.1, the STL file can be constructed in 2 formats (binary and ASCII format). Either format consists of the vertices of each facet in the model. These vertices can be read into a vertex list in which each vertex is represented by x, y, z coordinates. Each triangle facet is stored by the references in the vertex list and its normal vector. The facets and vertices lists provide a more efficient way to specify a patch. For

example, consider the following patch shown in figure 3.3. It is composed of 2 triangular facets defined by 4 vertices. Once the STL file is read into the patch representation, some



F_1	V_1	V_2	V_3	N_1
F_2	V_2	V_3	V_4	N_2

V_1	X_1	Y_1	Z_1
V_2	X_2	Y_2	Z_2
V_3	X_3	Y_3	Z_3
V_4	X_4	Y_4	Z_4

Figure 3.3: Patch representation.

preprocessing needs to be done preparing for the slicing procedure. First, the model needs to be moved to the x-y plane. This can be done by subtracting the minimum z value from the z values of each vertex:

$$z_i = z_i - z_{min} \tag{3.1}$$

where z_{min} is the minimum z value of the model. The model should also be moved to the center of the printing plate. This can be accomplished by moving the geometric center to the plate center, which is roughly determined by the mean of the x, y values of the vertices:

$$x_i = x_i - x_{mean}, y_i = y_i - y_{mean} \tag{3.2}$$

where (x_{mean}, y_{mean}) is the geometric center coordinate.

3.2 UTR, VTR and Dividing Planar

3.2.1 Dividing Planar

Once the model is grounded and moved to the center of the printing plate, it is ready to be sliced. In this step, the STL model is divided into two distinct types of regions: Uniform Thickness Region (UTR) and Variable Thickness Region (VTR). UTR is defined

as those regions where layer thicknesses are uniform while VTR is defined as those regions in the part where layer thickness can change. VTR regions are designated to minimize areas where the stair step effect is prevalent (i.e. areas with steep inclines). This proposed algorithm is to improve the surface finish by slicing the most suffered region with non-planar layers. These regions are determined as VTRs, and the rest of the 3D model is sliced into flat layers and determined as UTRs. For example, the VTR region in a hemisphere model is the region at the top, and the rest of the model is the UTR region. According to the definition of the base case, the entire model should be manufactured with the same number of layers, that is, each layer consists of UTRs and VTRs. The first step of this proposed algorithm is to determine the z value of the dividing planar to divide the model into two types of regsions, i.e., UTR and VTR. Figure 3.4 demonstrates the UTR and VTR

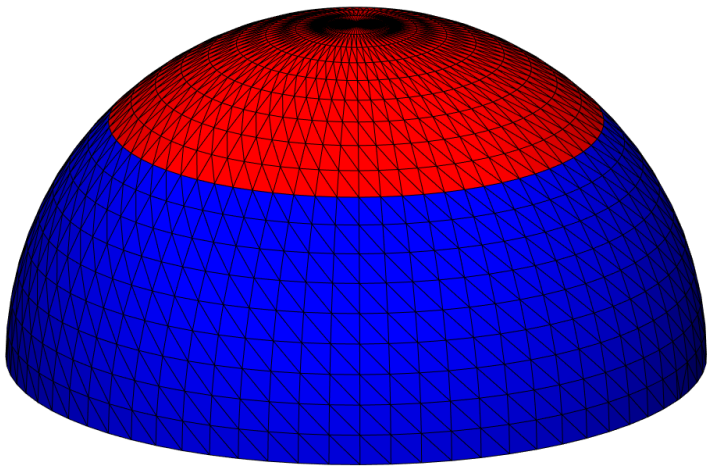


Figure 3.4: VTR and UTR in a hemisphere model.

regions in a hemisphere model. The red region is determined as the VTR and the rest, the blue region, is the UTR. To maximize efficiency, a threshold can be set to distinguish between URT and VTR:

$$F_{vtr} = \{f_i | z_{ni} > \tan\theta_{th}\} \tag{3.3}$$

where F_{vtr} is the set of the VTR facets, f_i is the facet, z_{ni} is the z value of the normal vector of f_i and θ_{th} is the threshold angle. In the example of the hemisphere, a threshold angle of 45° between the normal vector and the planar is set to distinguish between the UTR and VTR within a tessellated object, as shown in figure 3.4. In this hemisphere, the facets with normal vector angles greater than the threshold angle are facets at the top surface. Thus, in this step, the normal vectors of each facet in the STL model need to be examined.

In addition to the normal vector, there is another physical limitation; the layer thickness ratio δ, needs to be considered in this process. As the entire model should be manufactured with the same number of layers in the base case, in the same layer, the maximum ratio δ of the thickest available layer thickness to the thinnest available layer thickness is limited by the 3D printer's specifications. For most commercial desktop FDM printers, this ratio is around 3 (0.06mm - 0.2mm). In this research, the maximum thickness ratio δ is set to 3. This ratio limits the lowest possible z value of the VTR surface despite the z_{min}. The final z value of the dividing planar can be determined by:

$$z_d = \max(z_{min}, z_r) \tag{3.4}$$

$$z_r = \frac{z_{max}}{\delta} \tag{3.5}$$

where z_d is the z value of the dividing planar, z_{min} and z_{max} are the minimum and the maximum z value of the VTR facets. z_r is the lowest possible z value of the VTR surface.

3.2.2 Model Rebuild

By taking δ into consideration, the UTR and VTR are assured to be manufacturable. Once z_d is determined, the STL model needs to be rebuilt to be used for the layer generation. The STL model is cut into two facet sets, i.e., VTR set and UTR set. Figure 3.5 shows an example of a hemisphere cut by the dividing planar. Specially, the dividing planar cuts through some facets in the STL model and each of these facets must be divided into two or

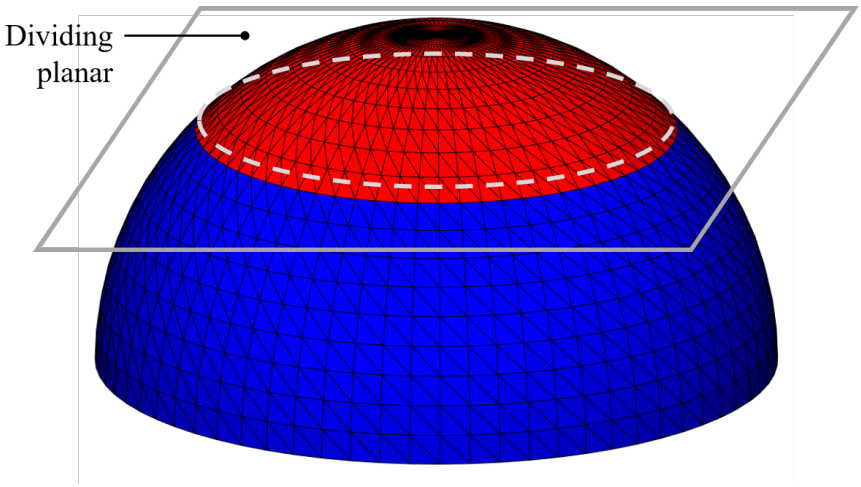


Figure 3.5: A hemisphere cut by the dividing planar.

three facets and be put in the VTR set and UTR set accordingly.

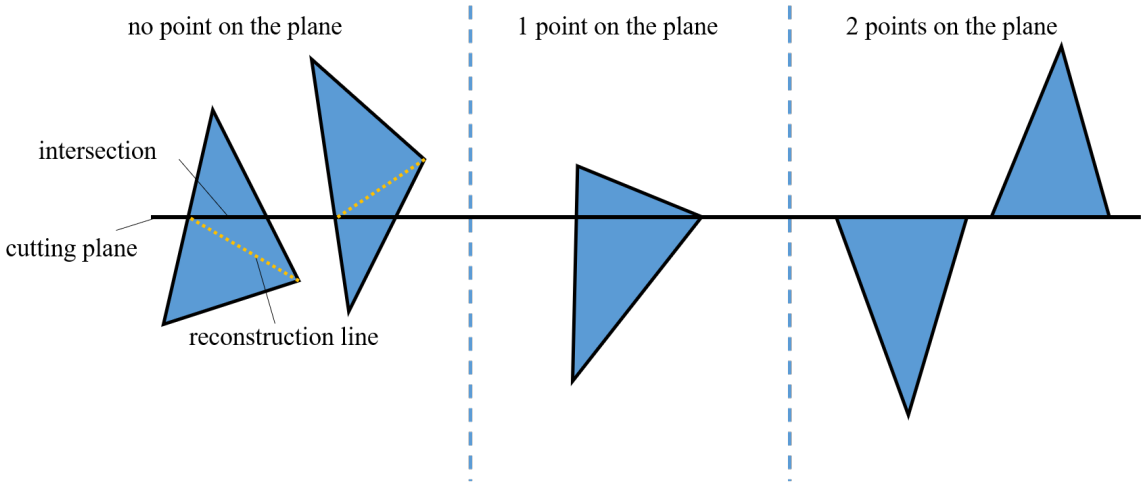
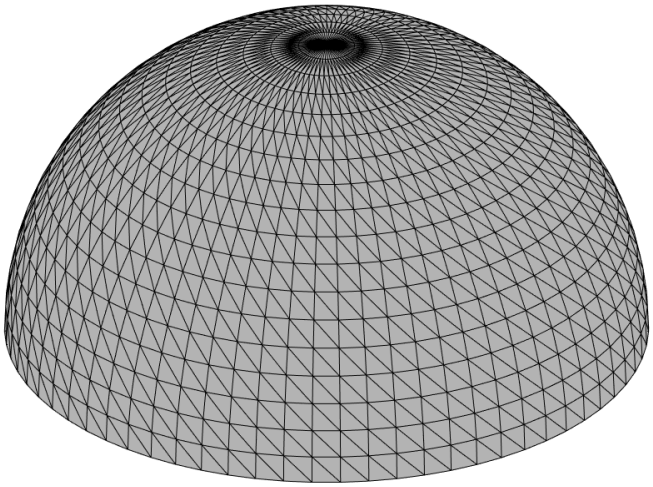


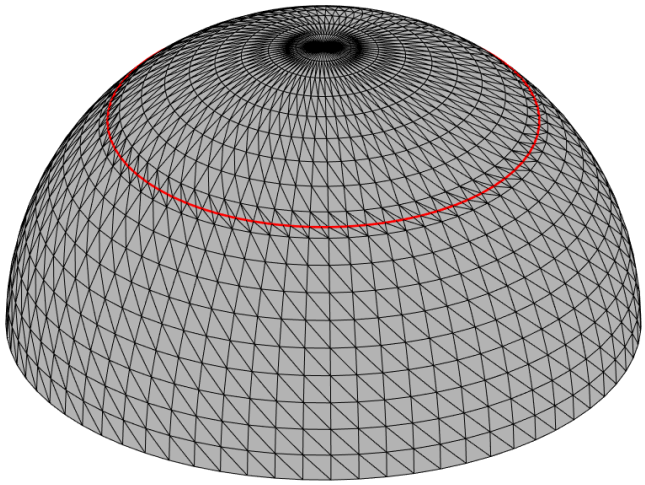
Figure 3.6: Different cases of the dividing planar cutting a triangular facet.

Figure 3.6 demonstrates the different cases of the dividing planar cutting a triangular facet. Intersecting triangles with no point on the cutting plane should be further segmented into three smaller triangles. Each triangle is divided into 3 sub-triangles by the intersection and the reconstruction line. The reconstruction line is the connection between the triangular vertex closest to the XY plane and its furthest intersect point. Intersecting triangles with one point on the cutting plane should also be naturally segmented into two sub-triangles, as shown in figure 3.6. No reconstruction line is needed for this case. If the dividing planar

is intersecting triangles with two points, no further segmentation is needed, as the planar is not cutting the triangle. By removing the original intersecting triangles and adding the sub-triangles in addition to the facet intersects and vertex list, the tessellated object remains the STL format for further processing. These reconstructed sub-triangles should keep the same normal vector as their original triangles.



(a) Before STL model rebuild



(b) After STL model rebuild

Figure 3.7: STL model rebuild.

Figure 3.7a and figure 3.7b show before and after the STL model rebuild of a hemisphere. The red contour in figure 3.7b is the cutting line of the dividing planar. The facets

that are cut by the cutting line are further divided into sub-facets in the STL model.

3.3 Slicing

Once the STL is rebuilt in the way introduced above, the STL model can be divided into UTR and VTR facet sets:

$$F_{vtr} = \{f_i | \min z_{ij} \geq z_d\} \tag{3.6}$$

$$F_{utr} = \{f_i | \max z_{ij} \leq z_d\} \tag{3.7}$$

where f_i is the facet, z_{ij} is the z value of one of the three vertices in f_i.

The number of layers and the UTR layer thickness can be determined by:

$$N = \left[\frac{z_d}{T}\right] \tag{3.8}$$

$$T_{utr} = \frac{z_d}{N} \tag{3.9}$$

where T is the desired layer thickness, $Tutr$ is the determined UTR layer thickness.

The contour of the VTR is also required for the model slicing procedure. This can be accomplished by the method introduced in previous section 1.3.1. In the example of the hemisphere, the perimeter of the VTR is the red line in figure 3.7b. Inspired by the uniform slicing algorithm introduced in section 1.3.1, this proposed slicing procedure cuts the model using cutting planes with different z values. The total height of the cutting plane is z_d, and the number of layers is N. Specifically, the ith UTR layer has the height of:

$$h_i = z_d \cdot \frac{i}{N} = T_{utr} \cdot i \tag{3.10}$$

where z_d is the z value of the dividing planar and $Tutr$ is the finalized layer thickness of UTR.

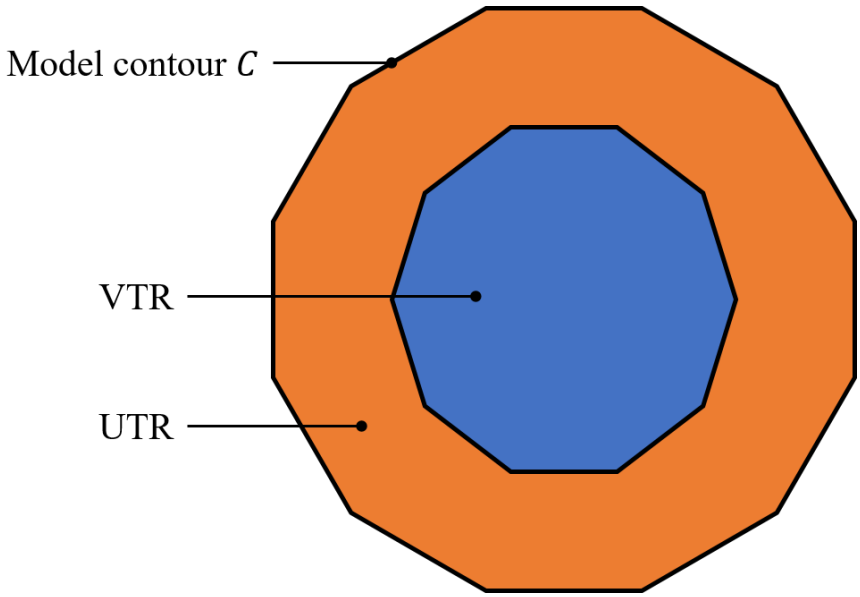


Figure 3.8: UTR and VTR in a layer.

Figure 3.8 demonstrates the UTR and VTR in one layer in the case of the hemisphere. For the ith layer, the model is cut by the cutting plane with the height of h_i. The perimeter of the STL model is represented by a polygon shown as the model contour C in figure 3.8. The perimeter of the VTR, which is extracted by the procedure introduced above, is shown as the contour of the blue area. Note that the polygonal perimeter is drawn exaggeratedly with a very small number of sides. In reality, the number of sides would be much larger. As shown in this figure, the perimeter of the VTR stays the same as in every layer, while the UTR changes with the cut contour of the model. In one layer, the UTR can be determined by using polygon clipping algorithms, i.e., Vatti clipping algorithm [56]:

$$P_{utr} = P_c \; not \; P_{vtr} \tag{3.11}$$

where P_{utr} is the perimeter of the UTR, P_c is the perimeter of the model in a layer and P_{vtr} is the perimeter of the VTR. This is a also known as one of the Boolean operations on polygons. As shown in figure 3.8, the VTR is represented in orange. Now the UTR is determined in a layer represented by a clipped polygon, while VTR cannot be represented

by a polygon, as it has variant layer thickness. The VTR is determined for each layer by offsetting the vertices and facets from the top surface by a certain distance, the z value of each vertex can be calculated by equation 3.12:

$$z_{ij} = i \cdot \frac{z_{tj}}{N} \tag{3.12}$$

$$x_{ij} = z_{tj} \tag{3.13}$$

$$y_{ij} = y_{tj} \tag{3.14}$$

where (x_{ij}, y_{ij}, z_{ij}) is the coordinate of jth vertex in the VTR of the ith layer and (x_{tj}, y_{tj}, z_{tj}) is the corresponding vertex in the VTR of the top layer. Specifically, at the perimeter of the VTR, the z value should be the same as the z value of the UTR in the same layer. Figure 3.9 shows the slicing result of a hemisphere with exaggerated layer thickness.

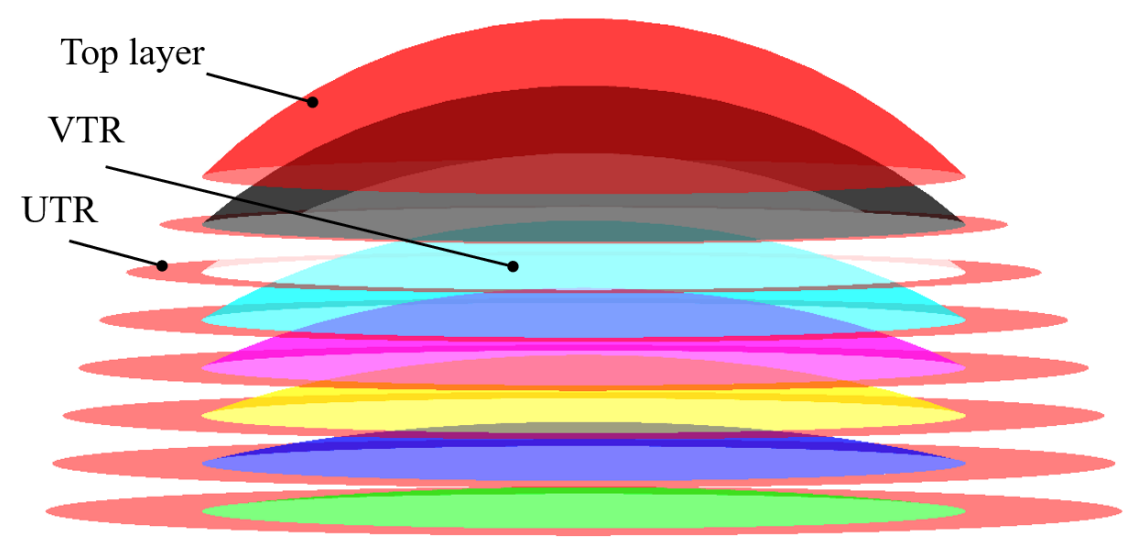


Figure 3.9: Slicing result of a hemisphere.

To be used in the further steps (i.e. toolpath generation), the slicing results need to be stored in a data structure. The data structure used to represent layers in this research is shown in figure 3.10. Each layer consists of the layer height z_{utr}, the layer's contour, represented as a polygon, the VTR contour, represented as a polygon, UTR polygons and

VTR facets.

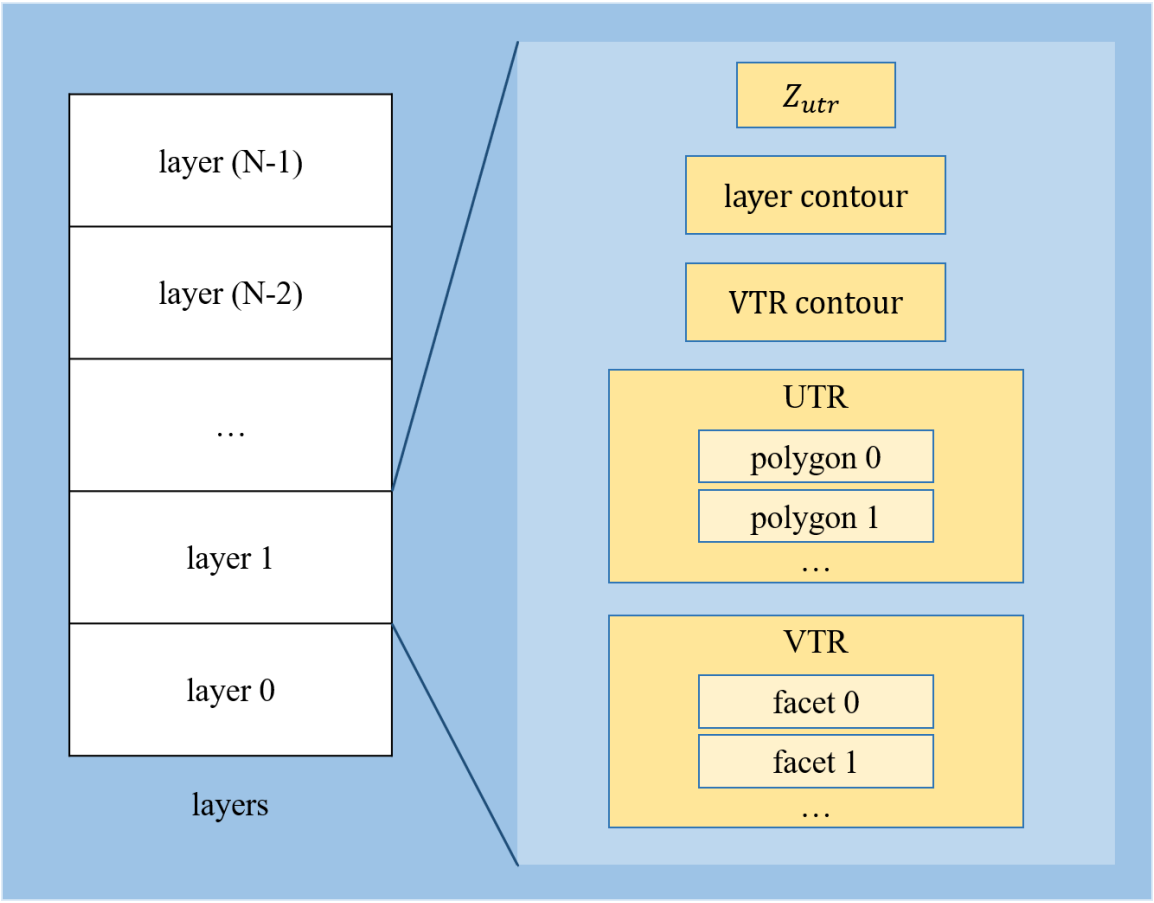


Figure 3.10: Data structure to store the sliced layers.

3.4 Path Planning

Once the model is sliced as demonstrated above, the slicing results are moved to the next stage of the whole process, i.e. path planning. This stage is to transform the layer information into printing paths that a 3D printer can follow. The stage contains several steps as shown in figure 3.11. First, the raft needs to be generated before the actual printing part; Second, for each layer, the inset paths and infill paths are generated; Then, the paths within each layer are ordered and the paths between consecutive layers are connected; Finally, the traveling and retraction movements are added to the plan. The path planning process in this research is inspired by the ”pipeline” process [57] proposed by *CuraEngine*.

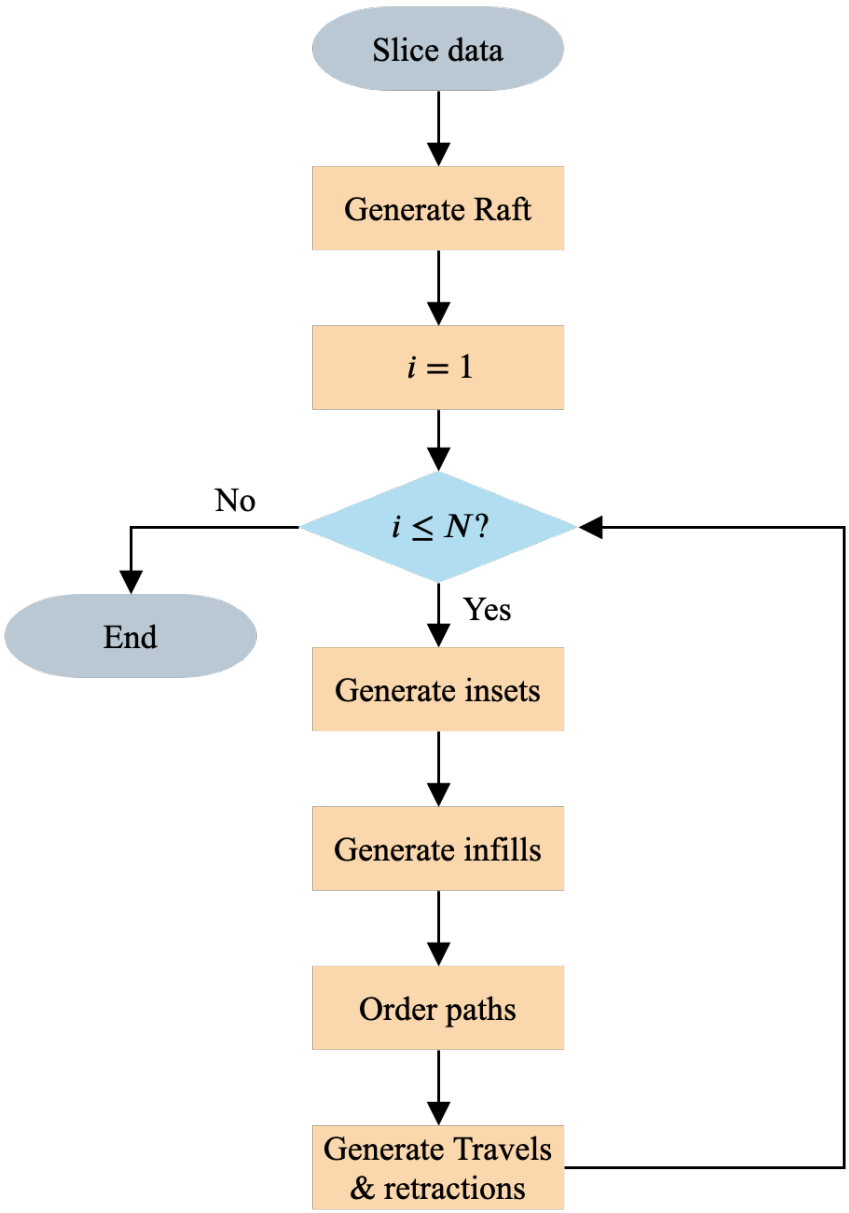


Figure 3.11: Flowchart of path planning.

3.4.1 Settings

In a 3D printing process, the print features and process parameters are controlled by the settings. For example, the initial nozzle temperature and the build plate temperature, the infill patterns, the layer thickness, etc. The implementation of storing and reading the settings in this research is inspired by *CuraEngine*. The settings are handled by storing in a JSON [58] file. JSON is a standard file format to store and exchange readable data

objects. It consists of key-value pairs and array data structures. An example of a JSON file is shown in Listing 3.1. In this example, the ”book” keyword’s value is an array of objects. In this research, the printing settings are stored in such format mapping from keys to setting instances. Each of the setting instances contains all of the attributes of a setting. The setting structure is hierarchical. For example, ”raft_top_speed” is under ”raft_speed”. In another words, the entire setting structure can be treated as a tree data structure [59]. When recalling a specific setting from the JSON setting file, a depth-first search (DFS) algorithm [60] is used to traverse the setting tree.

Listing 3.1: A JSON example

```
{
    "book": [
                {
                        "id": "01",
                        "language": "Java",
                        "edition": "third",
                        "author": "Herbert Schildt"
                },
                {
                        "id": "07",
                        "language": "C++",
                        "edition": "second",
                        "author": "E.Balagurusamy"
                }
        ]
}
```

3.4.2 Raft Generation

A raft is a thick grid between the model and the build plate. As the build plate used in this research is not perfectly flat, adding the raft is useful to make sure that the model will stick better to the build plate. The raft structure is visualized in figure 3.12. The floor plan of the raft can be confined by offsetting the first layer of the sliced model outwards by a certain raft extra margin, which is in the settings. The offsetting can be done using polygon offsetting algorithms [61] with two major steps: 1. Compute the offset by shifting each polygonal edge by the offsetting distance d away from the polygon; 2. Each pair of adjacent offset edges is connected by a circular arc of radius of d. In addition, if the polygon is a non-convex polygon, before these two steps, the polygon needs to be decomposed into convex sub-polygons. Then the two steps are repeated for each of the sub-polygons.

There are three parts in a raft: top layer, middle layer and bottom layer. The bottom layer is meant to be pressed against the build plate to get a better adherance, therefore it is usually designed to have a larger layer width. The middle layer is added upon the bottom layer to make more surface area for the top layers to lay down upon. Normally, there is only one middle layer. The top layers are the layers that contact with the model. This is meant to make the top surface of the raft as flat and smooth as possible; for that reason, the top layers work better with 100% infill.

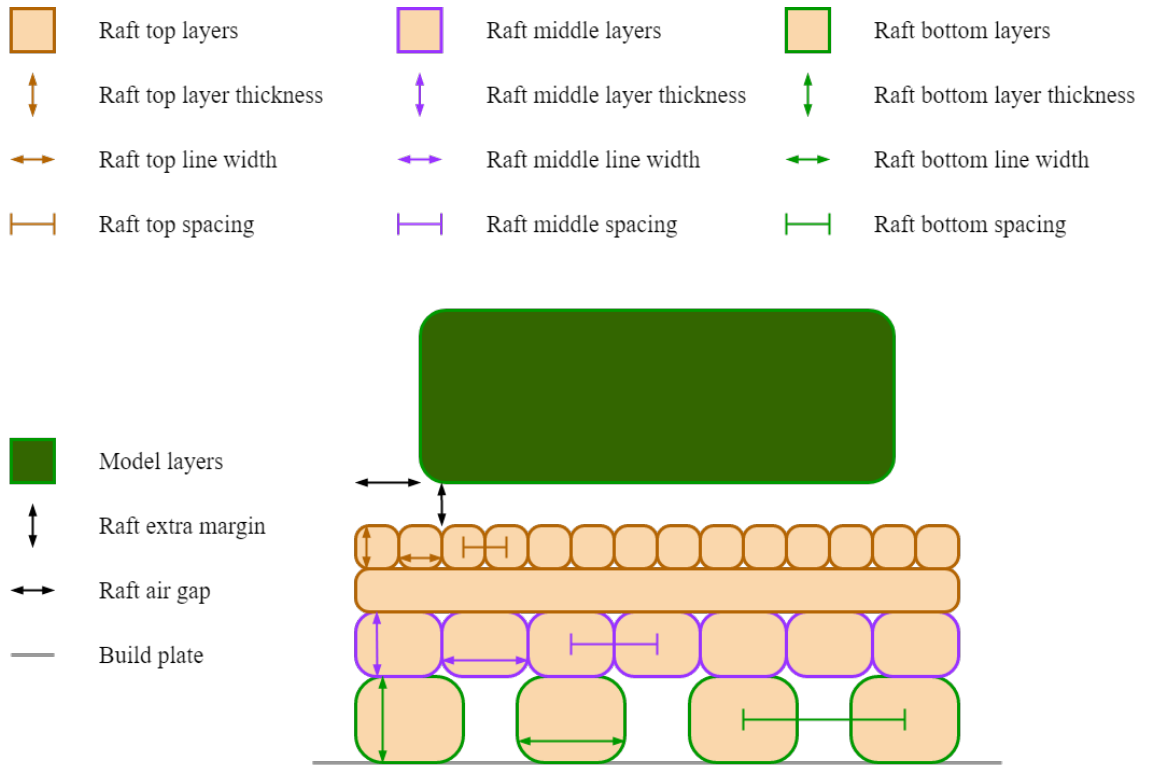


Figure 3.12: The raft structure.

3.4.3 Wall Inset Generation

After the UTRs and VTRs of the model are generated for each layer from section 3.3, they will be subdivided into zones that are designated to be filled to serve a certain purpose. In this research, each cross section of the model is divided into two kinds of zones, i.e. walls and infills. The first step to separate a cross section into zones is to isolate the parts that are going to become walls. In this step, a few insets are generated from the contour polygon of the layer, one for each wall. Figure 3.13 demonstrates the walls of a star-shaped layer. In this example, three walls are generated. One is the outer wall colored in red, and two are the inner walls colored in green. The outer most black star is the contour of the layer.

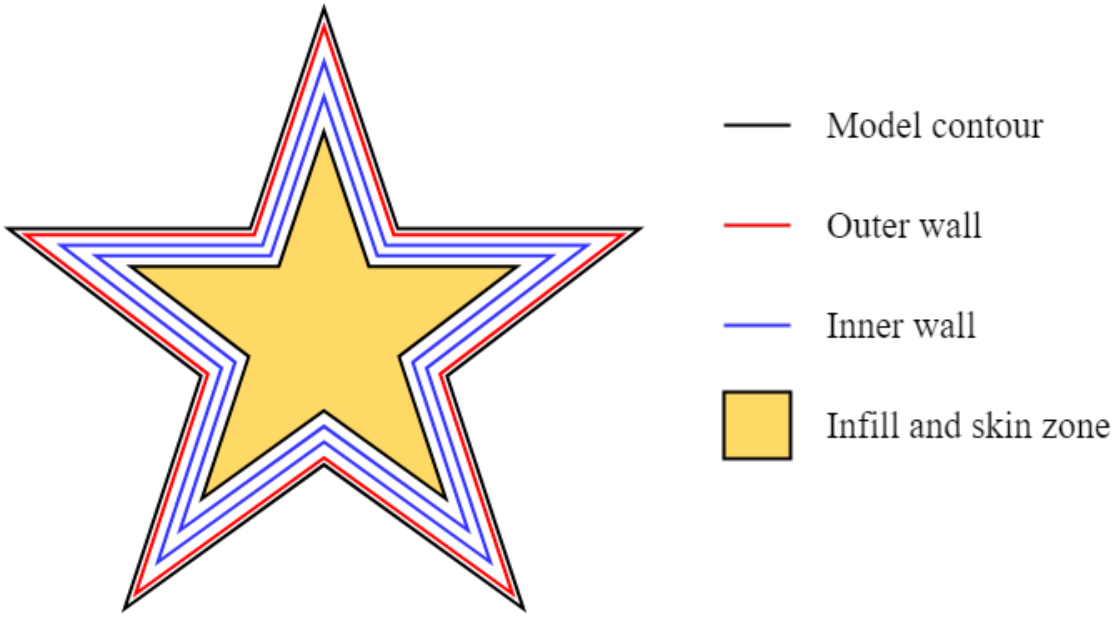


Figure 3.13: Demonstration of walls of a layer.

The outer wall inset is generated by offsetting with half of the outer wall line width of the contour of the model. The result is a polygon that goes through the middle of the outer wall. The vertices of this polygon will end up in the G-Code as the destination coordinates that the nozzle is moving towards.

The first inner wall inset then is generated by offsetting the outer wall polygon. This inset's offset distance is equal to half of the outer wall line width plus half of the inner wall line width. The half of the outer wall line width is on the inside half of the outer wall and the half of the inner wall line width is on the outside half of the first inner wall.

Finally, The second inner wall inset or any further inner wall insets are generated by offsetting the first inner wall polygon. The insets' offset distance from the previous inset is equal to one inner wall line width.

3.4.4 Infill and Skin Generation

After the wall insets are generated for a layer, the rest of the cross section needs to be filled with skin and infill as shown in figure 3.13, colored in yellow. Therefore, it needs to be

determined what area is going to become skin and what is going to become infill. Skin is the area that is contacting the air directly on either the upper or lower layer, while infill area is everywhere that is neither the wall nor the skin area. Skin areas are going to be 100% fill while infill areas are going to be filled with desired infill pattern.

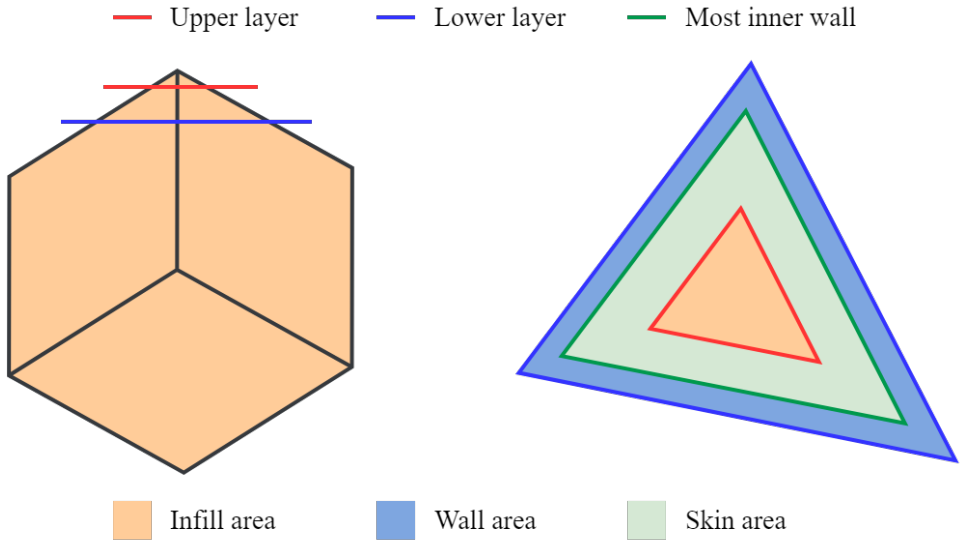


Figure 3.14: Infill and skin areas.

Figure 3.14 displays a demonstration of infill and skin areas. In this example, two consecutive layers, marked in red and blue, are sliced upon a rotated cube. The triangles on the right are the cross sections of these two layers. The blue and red triangles represent the lower and upper layers correspondingly. In the lower layer, the blue area represents the walls and includes all outer and inner walls. The green area that is between the walls and the upper layer is contacting the air, thus, this area is determined as skin. The rest of the area marked in orange then is determined as infill.

The reality of determining skin and infill is slightly more complex. Each layer is not only looking for its neighbor layers but also the layers that are within a skin thickness. For instance, if the skin thickness is set to be 2mm, and the layer thickness is 0.2mm, the layer is going to look for $2/0.2/2 = 5$ layers both upwards and downwards.

Once the skin and infill areas are determined, the infill line segments are next generated. The infill pattern can be selected by the user. The common infill patterns include grid, lines, triangles, cubic and zig-zag.

The most basic infill pattern is the line. Most infill patterns can be generated by lines. For instance, triangular infill is just a combination of 3 sets of lines with 60° angles from each other. In this research, grid infill is used for all the experiments for best practice and efficiency.

For linear infill types (i.e. lines, grid, triangles or zig-zag), the angle of the infill lines can be chosen by the user. Instead rotating the infill lines by the fill angle, rotating the model itself would be more efficient, such that the infill lines can keep axis-aligned.

Once the model is rotated by the fill angle, the next step is to determine the actual area within which to generate infill line segments. The actual infill area is known as the axis-aligned bounding box (AABB) of the model. The AABB can be determined by a min and max vector representing minimal and maximal coordinates in the x-y plane, i.e. (x_{min}, y_{min}) and (x_{max}, y_{max}). These two vectors can be obtained by examining every vertex in the model. If a vertex is out of the bounding box, then the bounding box is updated such that the vertex is included:

$$x_{min} = \min(x_{min}, x_i) \tag{3.15}$$

$$x_{max} = \max(x_{max}, x_i) \tag{3.16}$$

$$y_{min} = \min(y_{min}, y_i) \tag{3.17}$$

$$y_{max} = \max(y_{max}, y_i) \tag{3.18}$$

where (x_i, y_i) is the x, y coordinate of the vertex.

Figure 3.15 demonstrates the grid infill line segments of a 6-point star infill area with fill angle of 45°. The original infill area is colored in orange on the right side. To fill this area with 45° infill lines, the model is rotated by 45° so that the infill lines are aligned with

the axes. The black box in this figure is the AABB of the infill area. The dashed lines are the infill scan lines. The distance between the scan lines is calculated such that the line width of the infill pattern achieves the desired infill density. For example, To achieve 100% density, the line distance should be equal to one line width, twice the line width for 50% density, etc.

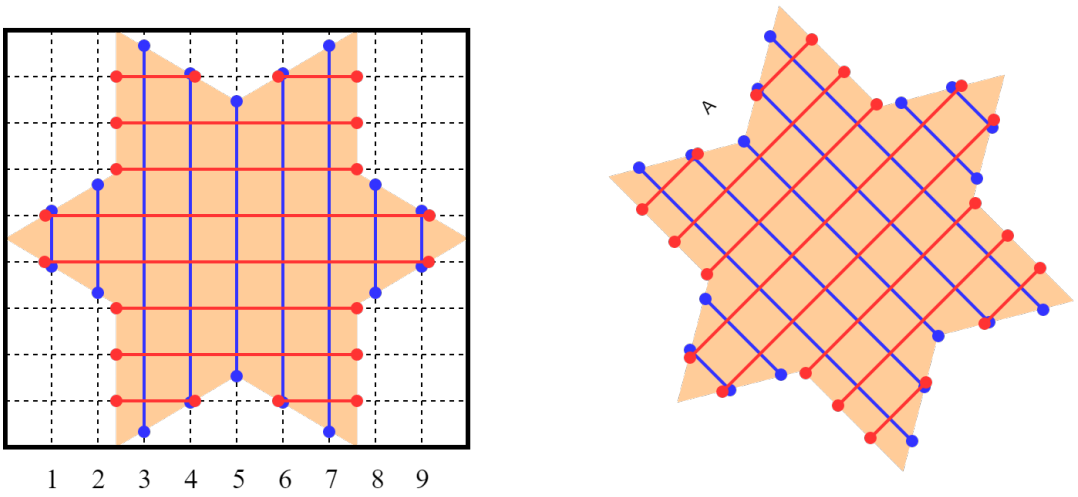


Figure 3.15: Grid infill line segments with fill angle of 45°.

Once the equal spaced infill scan lines are generated, the next step is to calculate all intersection points where the scan lines intersect the polygon of the infill area. Infill line segments are generated between each pair of two intersection points. In figure 3.15, infill line segments along the x and y axes are red and blue lines marked with oval arrows correspondingly. The algorithm to find out all the infill line segments is shown as follows. Note that this algorithm is finding all the infill line segments that are parallel to the y-axis. Infill lines that are on other directions can be determined by using the previously mentioned rotation technique.

First, index all the infill scan lines from the minimum x to the maximum x. As shown in figure 3.15, the vertical infill scan lines are indexed from 1 to 9. Initialize an

array,$A_i = []$, for each scan line index which stores the y coordinates of the intersection points on this scan line; Then, iterate through every edge of the infill area polygon (i.e. $l_i : [(x_{i1}, y_{i1}), (x_{i2}, y_{i2})]$, $x_{i1} < x_{i2}$). Find out all the infill scan line indices between x_{i1} and x_{i2}. Calculate the intersection points where the edge is intersecting with the scan lines and store the y coordinates into the array corresponding to the scan line index; Lastly, once all the edges of the polygons are examined, sort the y values in each array such that $A_j = [y_{j1}, y_{j2}, ..., y_{jn}]$ where $y_{j1} < y_{j2} < ... < y_{jn}$. Because polygons are closed shapes, the number of y values, n, must always be an even number. And the infill line segments can be determined that start from the odd indices and end at the next even indices:

$$S_i = \{[(x_j, y_{j1}), (x_j, y_{j2})], [(x_j, y_{j3}), (x_j, y_{j4})], ..., [(x_j, y_{j(n-1)}), (x_j, y_{jn})]\} \tag{3.19}$$

where x_j is the x coordinate of ith

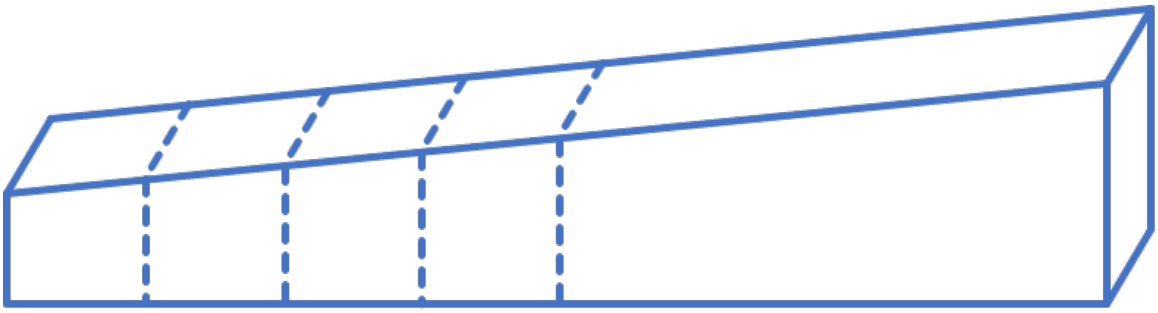


Figure 3.16: A VTR segment is divided into sub-segments.

In this research, the infill areas consist of both VTRs and UTRs. In most cases, the UTRs are usually surrounded by VTRs. Unlike the UTRs, the line thickness in VTRs , which directly affects the extrusion amount, is continuously changing along an infill line segment. Such segment will end up in multiple lines of G-Code in the later stages. Therefore, the infill line segments in VTRs should be divided into sub-segments such that for each sub-segment, the line thickness is approximated to be constant. Figure 3.16 demonstrates an infill line segment in VTR divided into sub-segments. To keep track of the thickness of each segment, the thickness needs to be put in the data structure of the segment, as pre-

sented in table 3.1. The type of an infill line segment in this research can be one of the following: *Infill*, *Inner Wall*, *Outer Wall*, *Skin* and *travel*. The $retract$ parameter indicates if a retraction of the filament is needed after current infill line segment. The thicknesses of the segments that are in UTRs are the same with the UTR layer thickness which is determined in section 3.3 as T_{urt}, while the infill line segments' thickness in VTRs needs to be determined by the average of the thickness at the start point and the end point. The process of determining the infill line segments in VTRs is as follows.

Table 3.1: Data structure of the infill line segment.

data	type	description
x_0	double	start point
y_0	double	
z_0	double	
x_1	double	end point
y_1	double	
z_1	double	
t	double	segment thickness
$type$	enum	segment type
$retract$	boolean	if retract after this path

First, through the same process as the UTRs to get the infill line segments in VTRs as the boundaries of VTRs are polygons just like UTRs; Then, each VTR infill line segment is divided into sub-segments with the length of 0.1mm. For each sub-segment, the triangular facets are found in the x-y plane where the sub-segment's start and end point land; Lastly, the thickness of each sub-segment is calculated by averaging the thicknesses at the start and end points. Figure 3.17 illustrates an example of a sub-segment. The start point p_1 and end point p_2 are located in two triangles.

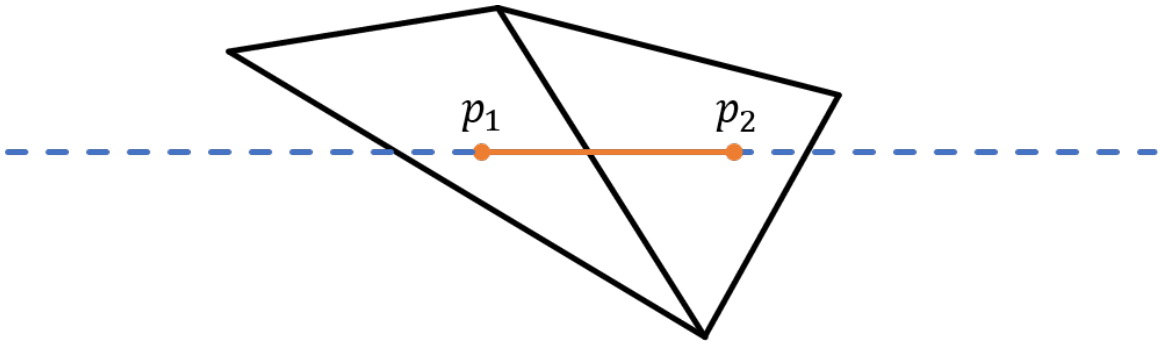


Figure 3.17: A VTR infill line sub-segment.

To determine which triangular facet the start and end point are located at, a computationally efficient algorithm is used in this research which is graphically demonstrated in figure 3.18. Point P can be represented as:

$$P = A + \omega_1 \cdot \overrightarrow{AB} + \omega_2 \cdot \overrightarrow{AC} \tag{3.20}$$

$$P_x = A_x + \omega_1(B_x - A_x) + \omega_2(C_x - A_x) \tag{3.21}$$

$$P_y = A_y + \omega_1(B_y - A_y) + \omega_2(C_y - A_y) \tag{3.22}$$

thus, we can have:

$$\omega_1 = \frac{A_x(C_y - A_y) + (P_y - A_y)(C_x - A_x) - P_x(C_y - A_y)}{(B_y - A_y)(C_x - A_x) - (B_x - A_x)(C_y - A_y)} \tag{3.23}$$

$$\omega_2 = \frac{P_y - A_y - \omega_1(B_y - A_y)}{C_y - A_y} \tag{3.24}$$

Point P is inside $\triangle ABC$ if:

$$\begin{aligned} \omega_1 &\geq 0, \\ \omega_2 &\geq 0, \\ (\omega_1 + \omega_2) &\leq 1 \end{aligned} \tag{3.25}$$

The triangles that contain the start and end point of a sub-segment can be determined by examining equation 3.25 for every triangle in the VTRs. However, this may not be efficient

enough, as there could be a massive number of facets in a VTR. To speed up this process, the AABB of each triangle is first tested for the points. The ω_1 and ω_2 then will be further examined only if the endpoint is located inside of the AABB of a triangle.

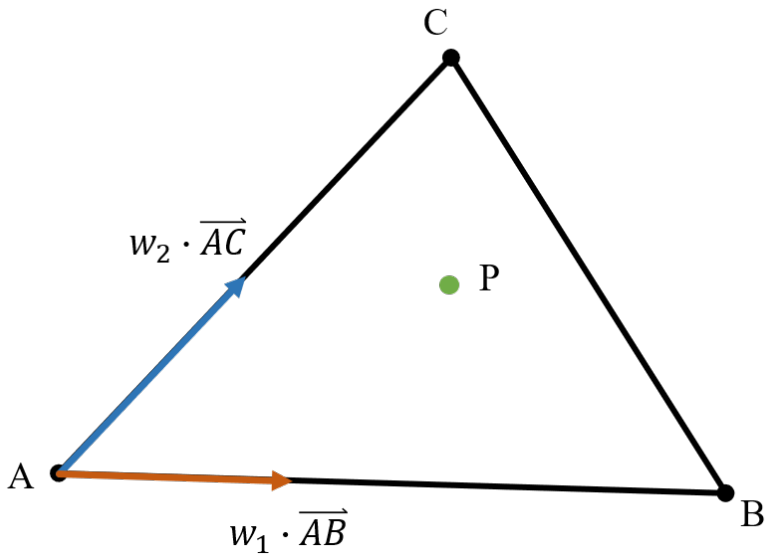


Figure 3.18: A computationally efficient method for determining if a point lies within a triangle.

Once the triangle is determined for an endpoint, the coordinate needs to be interpolated from the three vertices of the triangle. First, the plane equation can be obtained by the three vertices of the triangle just as a plane can be determined by three non-collinear points. Let $p_1 = (x_1, y_1, z_1)$, $p_2 = (x_2, y_2, z_2)$ and $p_3 = (x_3, y_3, z_3)$ be the vertices of a triangle and $ax + by + cz + d = 0$ be the plane equation. The normal vector can be obtained by:

$$\begin{aligned}\overrightarrow{n} = \overrightarrow{p_1p_2} \times \overrightarrow{p_1p_3} &= \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ (x_2 - x_1) & (y_2 - y_1) & (z_2 - z_1) \\ (x_3 - x_1) & (y_3 - y_1) & (z_3 - z_1) \end{vmatrix} \\ &= \mathbf{i}\left[(y_2 - y_1)(z_3 - z_1) - (y_3 - y_1)(z_2 - z_1)\right] + \mathbf{j}\left[(z_2 - z_1)(x_3 - x_1) - (z_3 - z_1)(x_2 - x_1)\right] \\ &\quad + \mathbf{k}\left[(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)\right]\end{aligned} \tag{3.26}$$

therefore, the plane equation can be determined by plugging in p_1 into equation 3.26:

$$\begin{aligned} &ax + by + cz + d = 0 \\ &a = (y_2 - y_1)(z_3 - z_1) - (y_3 - y_1)(z_2 - z_1) \\ &b = (z_2 - z_1)(x_3 - x_1) - (z_3 - z_1)(x_2 - x_1) \\ &c = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) \\ &d = -(ax_1 + by_1 + cz_1) \end{aligned} \tag{3.27}$$

Once the plane equation is determined, the z value of the endpoint can be easily obtained by plugging in the x and y values into the plane equation 3.27:

$$z_i = -\frac{ax_i + by_i + d}{c} \tag{3.28}$$

Thus, the thickness of the sub-segment can be determined by averaging the thicknesses at the start and end point:

$$t_i = \frac{z_{i1} + z_{i2}}{2N} \tag{3.29}$$

where z_{i1} and z_{i2} are the z value of the start and end point, and N is the number of layers.

3.4.5 Path Ordering

So far, all the infill line segments and sub-segments are generated and stored in the data structure described in figure 3.19. For each layer, the segments or sub-segments are categorized by their types. The next step is to order and connect all the segments together to generate toolpaths such that the corresponding G-Code can be generated in further steps.

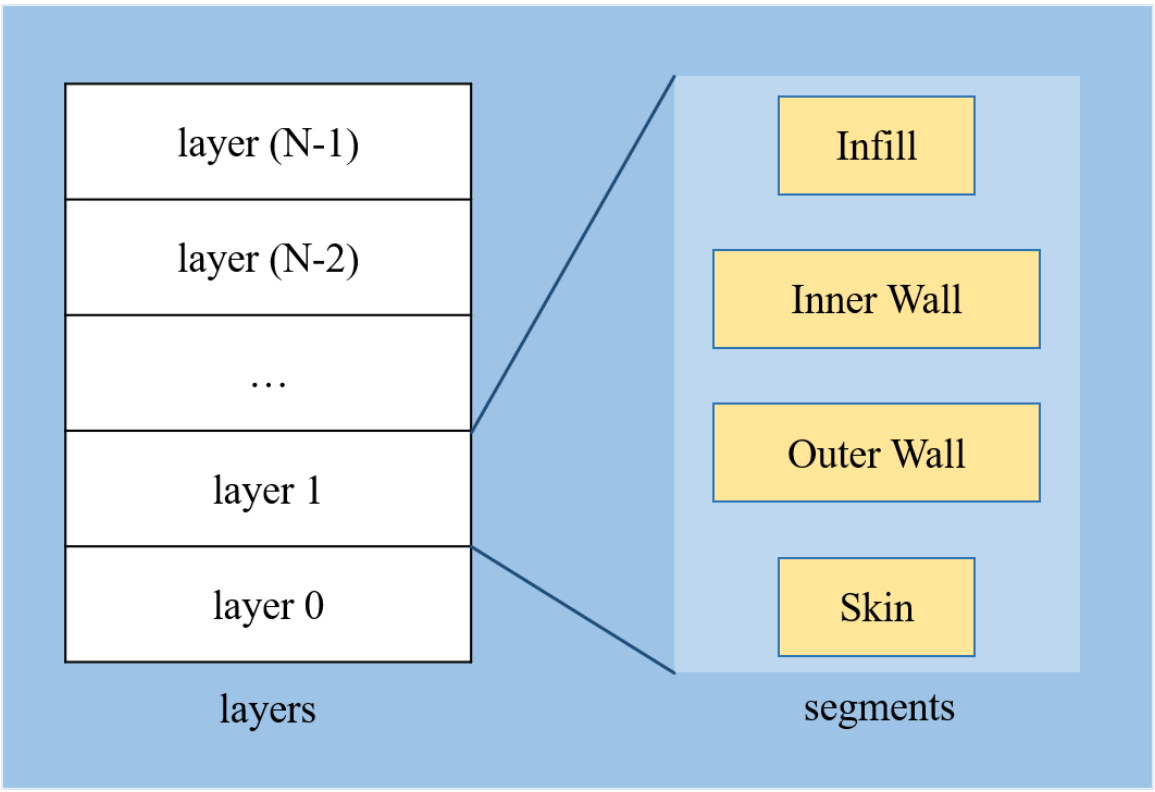


Figure 3.19: The data structure to store all the infill line segments for layers.

The paths are generated in the same order in which they will be printed. For each layer, the segments are printed in the order of: inner walls, outer walls, infills and skins. Different slicers handle this order differently. The reason behind this order is as follows. First, the reason why walls are printed before the infills and skins is that the walls give the infill support such that the infills have something to adhere to on the edge of the extrusion. Also, it is a geometrical constraint to keep the infills within, especially when the printer is moving at higher speeds. Second, the reason why the inner wall is printed before the outer wall is that when the parts have overhangs, a part of the outer walls is not printed on top of anything. The inner wall can provide extra support for the outer wall to stick to.

As the data structure has grouped these segments into different types by itself, the major task in this step is to connect the infill line segments in each group and add necessary traveling paths. When printing a bunch of lines with the same type together, the order in which to print these lines can be optimized approximately. Finding the optimal order

is a Traveling Salesman Problem [62–65], which is an NP-hard problem [66, 67]. Only exponential-time algorithms exist for this kind of problem. Apparently, the exact solution is infeasible for this application within a reasonable time span. Instead, the Nearest Neighbor Algorithm, which has the time complexity of $O(n^2)$, is implemented in this research. The pseudo code of this algorithm is shown in algorithm 1. The basic idea is to find the nearest endpoint of another infill line segment to the current one.

Algorithm 1 Nearest Neighbor Algorithm

1: Initialize all endpoints as unvisited
2: Select an arbitrary endpoint u_s as the start endpoint of the segment
3: **while** unvisited segment exists **do**
4: Set the end endpoint of the segment is u_e as visited
5: Find out the shortest distance between u_e and an un-visited endpoint v_s
6: Set v_s as visited
7: $u_s \leftarrow v_s$
8: **end while**

Algorithm 1 can not only connect all the infill line segments within the same type group, but also within the segment groups. For example, after the infills are connected, the end point of the infills and the start point of the inner wall can be connected using algorithm 1 by finding the nearest neighbor. It is worth mentioning that one of the universal problems for FDM 3D printing technology is the ”layer seam”, also called the ”z seam”. An example of a ”layer seam” is shown in figure 3.20. This is because the start and end position of the outer walls for each layer are lined up and located at the same spot. The movement of the extruder is not as smooth as the rest of the outer wall; therefore, the extruder leaves a defect at this point on each layer. There is no way to completely get rid of this flaw, but there are a number of ways to make the seams less noticeable. Wipe, coast and extra restart distance are commonly used in modern slicers. In this research, a random start point is chosen for the outer wall of each new layer in the z direction. In this way, the next layer's outer wall

starts at a random point instead of the nearest point from the previous path, which reduces the chance of a seam. Note that this method could marginally increase the total print time, as it increases the travel distance.

Figure 3.20: An example of layer seams.[68]

Once all the infill line segments are ordered, they need to be connected into paths, which include extrusion paths and traveling paths. The traveling paths should be added between the discontinuous extrusion path. Figure 3.21 illustrates an example of infill paths in a layer. The blue lines represent the infill lines, and each line is discontinuous from other lines, i.e., a travel path needs to be added between every infill path as colored in red arrow lines in the figure. Either an extrusion path or a traveling path can be stored in the data structure described in table 3.1. The *type* of the traveling paths is set to ”travel”. When the travel distance is larger than a threshold, which is 5mm in this research, the path is considered as a travel path with retraction.

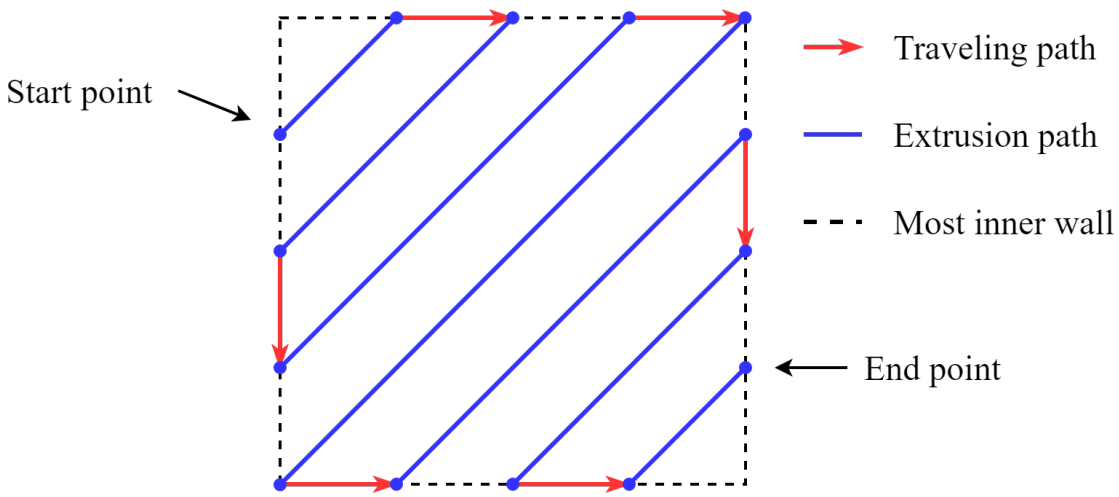


Figure 3.21: An example of infill paths in a layer.

When moving from one position to another, it is important to avoid hitting the printed part with a hot nozzle at the wrong spots, which is also known as interference. Figure 3.22 demonstrates a case wherein the nozzle needs to move up in the z direction before moving in the x-y plane. In this example, a traveling path is made from point A to B. The following steps are necessary to make this movement: First, the *retract* parameter in the previous path which ends at point A is set as *True*, which indicates a retraction will be performed after this path; Second, the maximum height of the current layer z_{max} is recalled to calculate the z value that the nozzle needs to move to using the equation $z = z_{max} + z_{hop}$, where z_{hop} is the z-hop height (in mm).

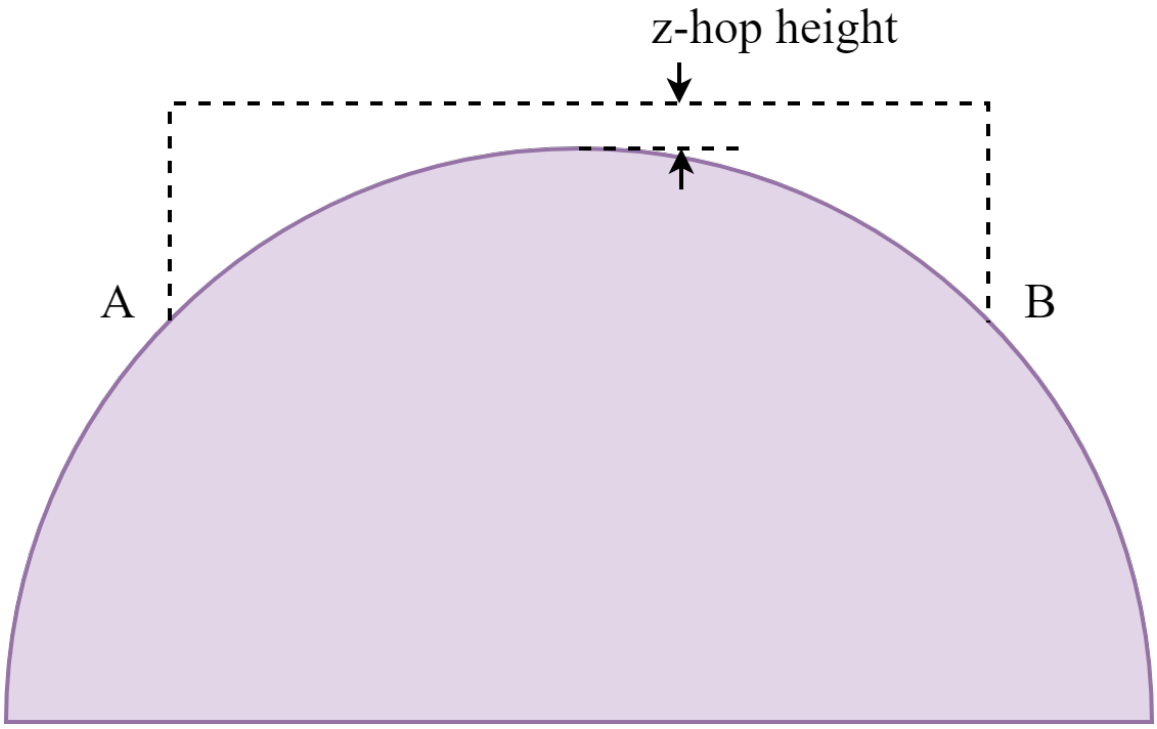


Figure 3.22: Moving over the part to avoid interference.

3.5 G-Code Generation

The final step of the slicing process is to export the G-Code based on the path data generated from the previous steps. First, the starting code that sets up the machine is added at the beginning of the G-Code file. These commands are listed in table 3.2; Then, the paths data are one-on-one mapped to G-Codes. In this step:

- A extrusion path having the same *type* parameter with the previous path is directly translated to a G1 command, followed by the destination coordinates and extrusion of the path as the parameters of the command.
- A travel path is translated to a G0 command, only followed by the destination coordinates.

Lastly, the end codes are added to the end of the G-Code file. These codes are meant to turn off the extruder and bed heaters (M104 and M140), retract the filament and move up

Table 3.2: Commands in start G-Code.

command	description
G21	set unit to metric
G90	absolute positioning
M82	put the E axis into absolute mode
M107	turn of the fan
M851	set the z probe offset
G28	auto home
M206	set home offsets
M206	set home offsets
G92	set position
M117	set LCD message

the nozzle (G1), and turn off the motors (M84).

The E parameter in each extrusion command (G1) indicates the amount of the material that needs to be extruded while making the movement. The extruded filament is modeled as a cuboid with the line width as the width, the path thickness as the height and the path length as the length. Therefore, the volume of the path extrusion path is:

$$v_i = t_i * w * l \tag{3.30}$$

where t_i is the path thickness, w is the line width and l is the line length.

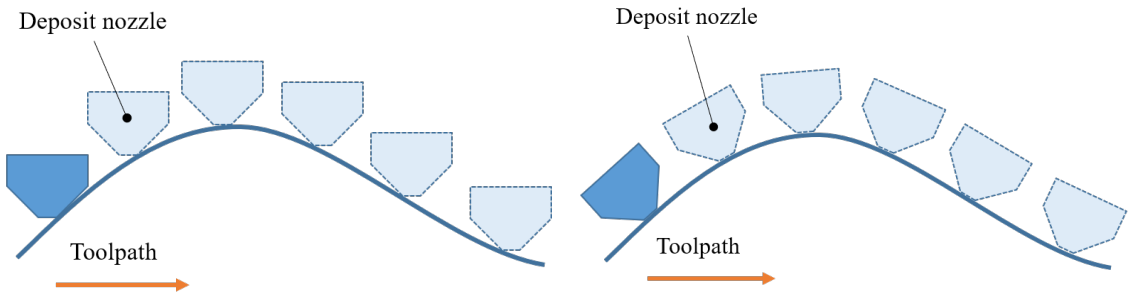
3.6 Discussion

3.6.1 Implementation on 5-axis FDM printer

Much research has been done to discuss the feasibility and design of multi-axis FDM systems. Wulle et al. identified some of the requirements of multi-axis AM and discussed possible directions of solutions to address the challenges [69]. Grutle made a 5-axis 3D printer from modifying a 3-axis FDM printer, showing that 5-axis system can achieve better surface finish with a shorter printing time [70]. Wu et al. developed an algorithm to

decompose models into support-free parts that can be printed with a 6-DOF robotic system [71].

In this research, to print the VTR, the ideal choice of the machine tool would be a 5-axis CNC machine to make the extruder axis be always normal to the point on the surface of deposition so that the extruded material can bond to the former layer by applying normal force from the nozzle tip. As shown in figure 3.23a, to applying CLFDM on a common 3-axis FDM machine, the extrusion axis is kept vertical through the printing process. Conversely, the extrusion axis is simultaneously adjusted to be normal to the surface when the nozzle moves along the extrusion path, shown in figure 3.23b. In addition, to facilitate a wider range of printable angles, the tip of extruder nozzle needs to be reshaped.



(a) FDM with 3-axis system

(b) FDM with 5-axis system

Figure 3.23: Demonstration of nozzle pose along tool path on 3-axis and 5-axis FDM.

3.6.2 Limitations of Base Case

As mentioned, a hemisphere is taken as an instance for the base case in this research. The STL model of a hemisphere is shown in figure 3.1. In this base case, only the top surface is considered in the model, which limits the applicability of the curved layers. In addition, the entire model needs to be manufactured by the same number of layers, which restricts the types of the models that are applicable to this algorithm. In chapter 4. these issues will be address and explained in detail.

3.7 Summary

This chapter introduces the entire slicing procedure for the base cases. First, the base case is defined with some constraints. Second, importing the STL model and some preprocessing is carried out. Third, the model is divided into VTRs and UTRs by a dividing planar. Then, the model is actually sliced into layers and stored in a customized data structure. After that, the paths are generated upon different types of insets (i.e. raft, walls, infills and skins). The paths are also ordered and connected using the Nearest Neighbor Algorithm. Lastly, the G-Codes for the paths are generated.

CHAPTER 4

LAYER THICKNESS RATIO INDEPENDENT SLICING PROCEDURE

A variable layer thickness slicing procedure for the base case is proposed in chapter 3. This chapter demonstrates a more generalized slicing procedure succeeding the procedure for the base case. This generalization is described in detail. Finally, the limitations of this procedure are discussed.

4.1 General Cases

In chapter 3, a variable layer thickness slicing procedure is introduced for the base case. The base case has the following constraints. There is no support structure allowed in the base case, which means the bottom surface of the model needs to be flat. In addition, all the VTRs and UTRs must have the same number of layers. This rule limits the maximum height ratio between the peak and the valley of the top surfaces that can be printed in VTRs, due to the limit of layer thickness that a printer can provide.

Figure 4.1 shows an exaggerated example of the model that the base case slicing algorithm is unable to address. This example show the VTR region of a model in 2D. In a single layer of the sliced model, of which the top and bottom surfaces are marked in red. The thickest spot and the thinnest spot in this layer are labeled as t_{max} and t_{min}. The following equation must be satisfied:

$$r_{max} > \frac{t_{max}}{t_{min}} \tag{4.1}$$

where r_{max} is the maximum layer thickness ratio that a FDM printer can produce. For most commercial desktop FDM printers, a typical range of layer thickness is 0.1-0.3mm, thus the r_{max} is about 3. If the layer thickness ratio of the model is greater than r_{max}, then the

VTR of the model needs to be shrunk down until the layer thickness ratio is smaller than r_{max}.

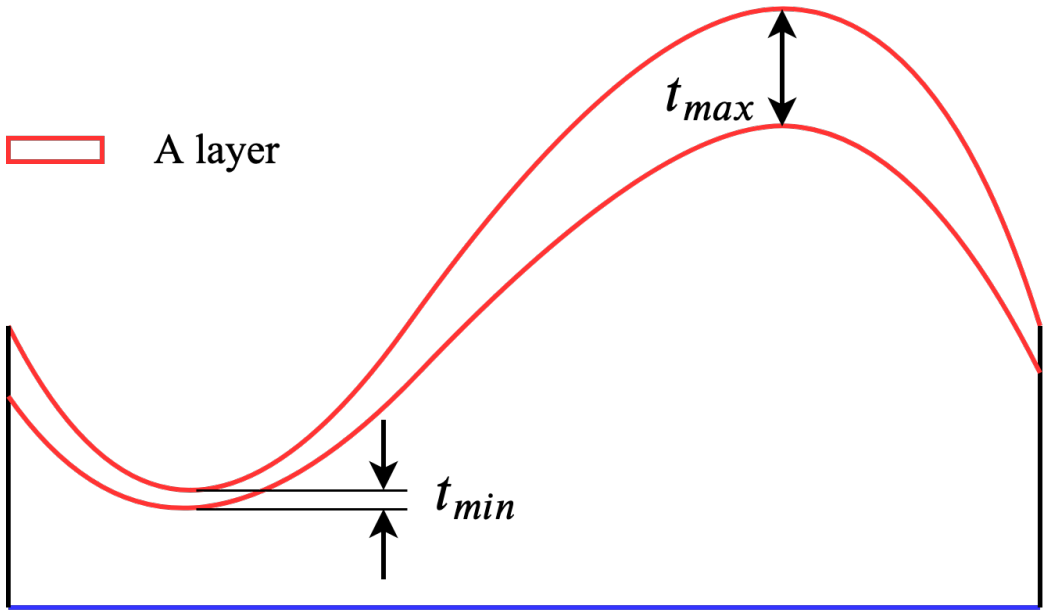
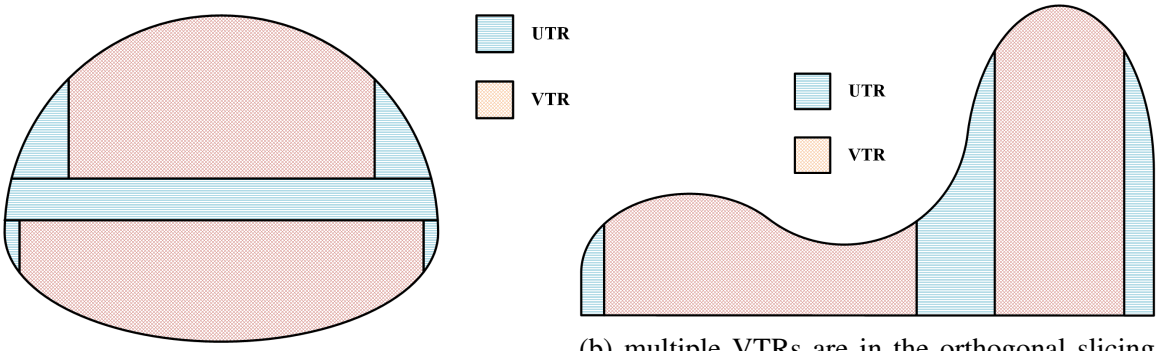


Figure 4.1: An example of the model that the base case slicing algorithm cannot address.

A more generalized slicing procedure is proposed in this chapter to address the layer thickness ratio limitation of the base case slicing algorithm. The generalization of the algorithm is presented in two sub-cases: sub-case 1 is the case in which multiple VTRs are in the slicing direction; sub-case 2 is the case in which multiple VTRs are in the orthogonal slicing direction and each VTR can have different number of layers.

Figure 4.2 shows the examples of the two sub-cases. In figure 4.2a, both the top surface and the bottom surface are curved. These two curved surfaces are converted into two VTRs and UTRs independently by two dividing planes. Then, the number of layers is determined accordingly for the VTRs and UTRs. Finally, the layers are generated by a similar method described in chapter 3. Figure 4.2b demonstrates sub-case 2 which has two VTRs with different numbers of layers, i.e., the VTR on the left has a smaller number of layers than the VTR on the right side. The two VTRs are connected by a UTR in the middle. In this case, the VTRs are determined by a method similar to that described in chapter 3. Then,



(a) multiple VTRs are in the slicing direction

(b) multiple VTRs are in the orthogonal slicing direction with different numbers of layers

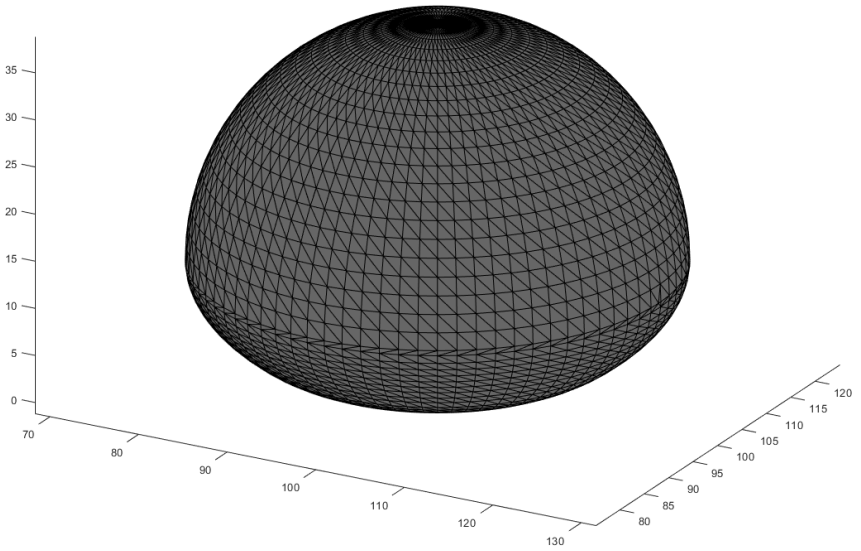
Figure 4.2: Demonstration of the two sub-cases.

the layers of the VTRs are determined independently. After that, the contour of the UTRs of each layer is calculated. Lastly the layers are generated.

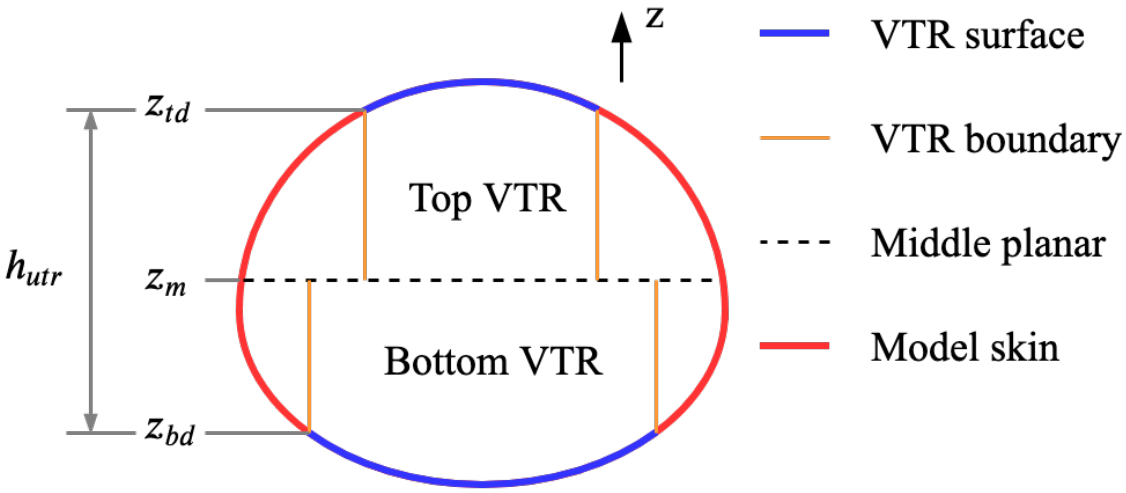
These two sub-cases decompose the model in two directions, i.e. the slicing direction, which is also known as the z direction, and in the orthogonal slicing direction, which is the x-y plane. By slicing the decomposed model with the two sub-case-slicing procedure, the entire model can be sliced into layers. In this research, a model that has both curved top and bottom surfaces is sliced to validate sub-case 1; a model that has two VTRs with different numbers of layers is sliced to validate sub-case 2. With the validation of these two sub-cases, the procedure can be generalized based on mathematical induction: a model that has both curved top and bottom surfaces, and has any number of VTRs with different numbers of layers, should be able to be sliced using the proposed method.

4.2 Sub-Case 1

Sub-case 1 is the case that has both curved top and bottom surfaces. The key to addressing this case is to determine the middle planar, which divides the model into two parts, i.e. the top part and the bottom part. Figure 4.3a presents an example STL model of sub-case 1. Figure 4.3b demonstrates the VTRs and UTRs in the model. As shown in the figure, two VTRs are determined for the top and bottom surfaces accordingly. The middle planar separates the two VTRs.



(a) multiple VTRs are in the slicing direction



(b) 2D demonstration of VTRs and UTRs in the model

Figure 4.3: Demonstration of the sub-case 1.

As mentioned above, the key step to slicing the model of sub-case 1 is to find the appropriate middle planar that separates the top and bottom VTRs. Before that, a preprocess step is used, identical to the one introduced in section 3.1. The STL model is read and re-

positioned to be grounded and centered. Then, the dividing planar for both top and bottom surfaces is determined through a similar process that is described in section 3.2.1:
First, the top VTR facets and the bottom VTR facets can be determined by a threshold angle:

$$F_{t\_vtr} = \{f_i | z_{ni} > \tan\theta_{th}\} \tag{4.2}$$

$$F_{b\_vtr} = \{f_i | z_{ni} < -\tan\theta_{th}\} \tag{4.3}$$

where F_{t_vtr} and F_{b_vtr} are the sets of the top VTR facets and the bottom VTR facets accordingly, z_{ni} is the z value of the normal vector of f_i and θ_{th} is the threshold angle. Then, the ratio of the number of layers for the top and bottom part, r_{t_b}, is determined by:

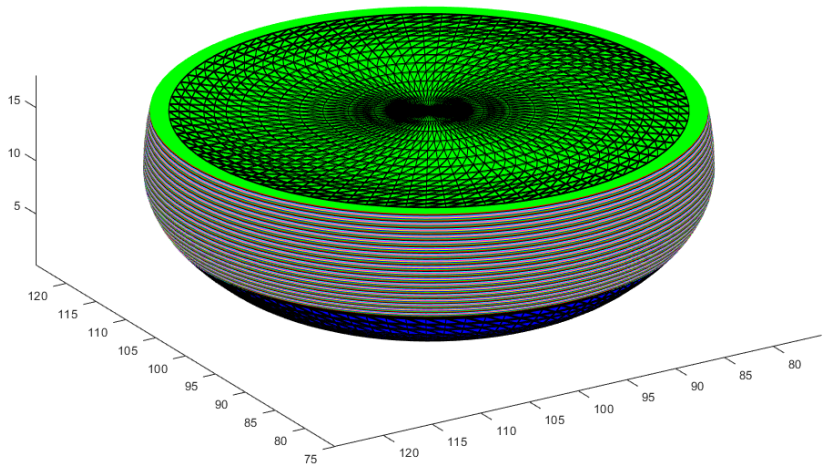
$$r_{t\_b} = \frac{z_{max} - z_{t\_min}}{z_{b\_max}} \tag{4.4}$$

where z_{max} is the maximum z value of the model, z_{t_min} is the minimum z value of the vertices in F_{t_vtr}, and the z_{b_max} is th maximum z value of the vertices in F_{b_vtr}. The next step is to determine the height of the UTR, h_{utr}. It can be determined by taking the layer thickness ratio δ into consideration:

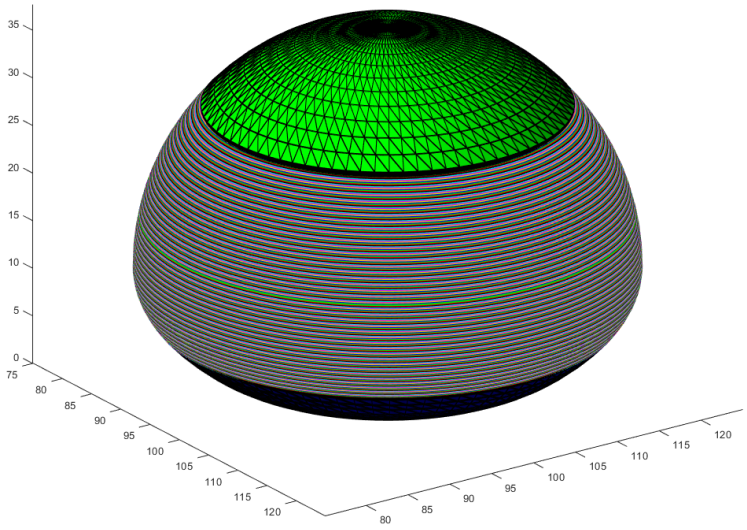
$$h_{utr} = \max\{h_{min\_utr}, (z_{t\_min}, z_{b\_max})\} \tag{4.5}$$

$$h_{min\_utr} = \frac{z_{max}}{\delta} \tag{4.6}$$

This assures that the maximum layer thickness ratio within any layer does not exceed the allowable thickness ratio δ that the 3D printer provides. Once the h_{utr} is determined, the z values of the dividing planes for the top VTR and the bottom VTR, denoted as z_{td} and



(a) Generated layers for the bottom part of the model



(b) Generated layers of the model

Figure 4.4: The slicing result of the example model of the sub-case 1.

z_{bd} accordingly, can be computed by:

$$z_{bd} = \frac{z_{max} - h_{utr}}{1 + r_{t\_b}} \tag{4.7}$$

$$z_{td} = z_{bd} + h_{utr} \tag{4.8}$$

After the dividing planes are determined for the top and bottom surfaces, the STL model needs to be rebuilt by using the method described in section 3.2.2. Finally, the layers are generated with the same procedure described in section 3.3. Figure 4.4 presents the slicing result of the example model. Note that the layer at the middle planar is flat, as shown in figure 4.4a.

4.3 Sub-Case 2

Sub-case 2 is defined such that the model has multiple VTRs with different numbers of layers. An example STL model used to demonstrate this case is shown in figure 4.6a. Figure 4.6 demonstrates the VTRs and UTRs in the model. As shown in the figure, two VTRs, VTR 1 and VTR 2, are determined for the higher and the lower curved top surfaces. These two VTRs cannot have the same number of layers because the height ratio exceeds the layer thickness ratio δ.

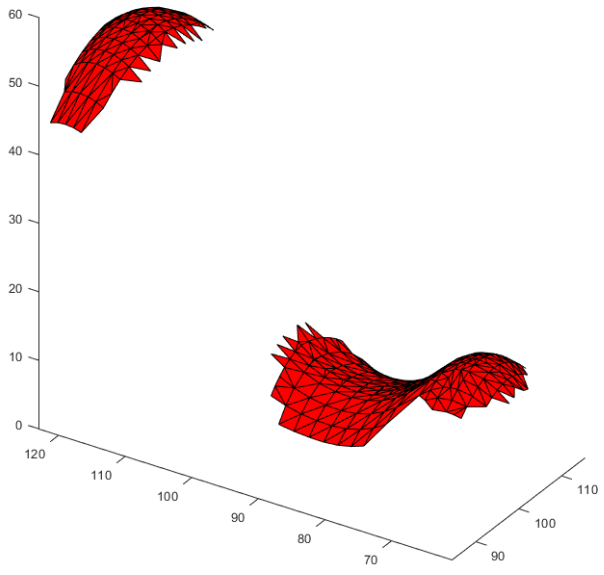
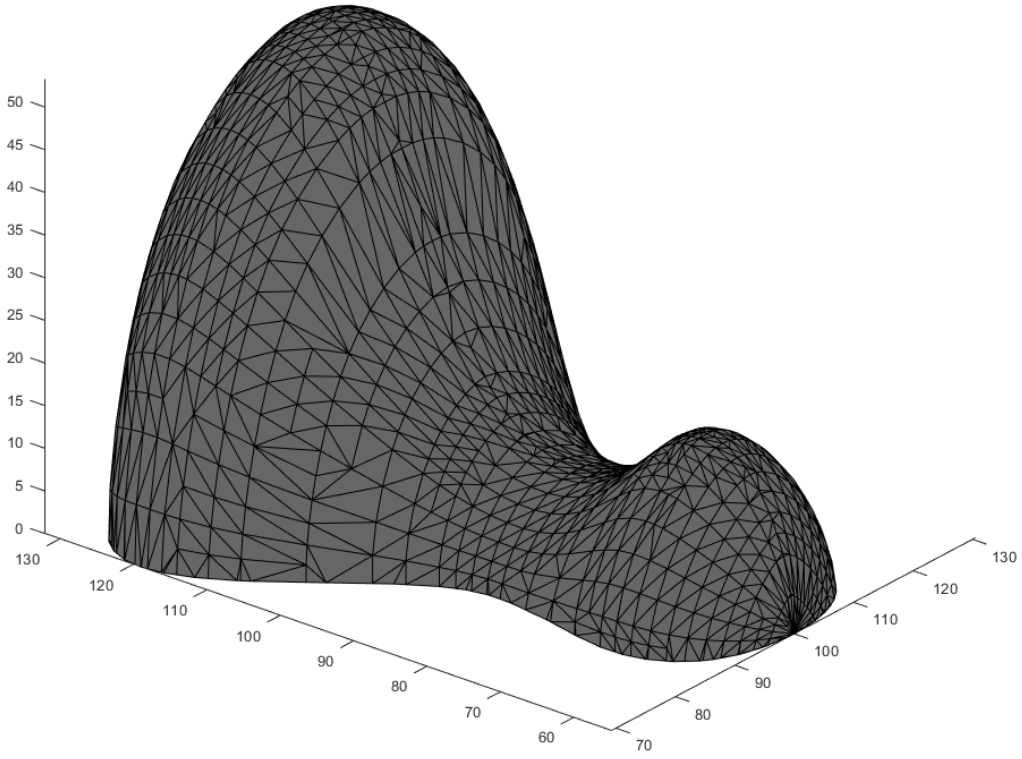


Figure 4.5: The extracted VTR facets by using threshold angles.

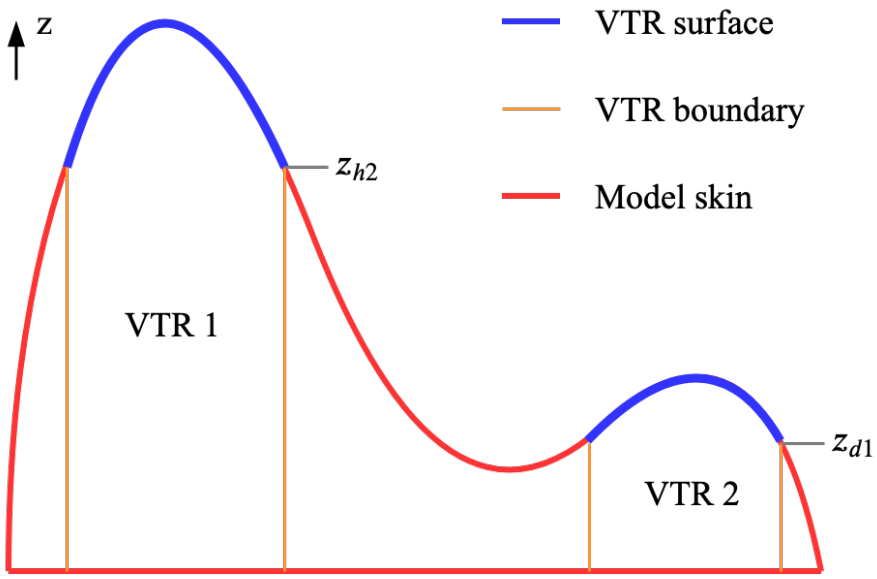
The STL model is read and re-positioned through the same preprocess described in section 3.1. Then, the VTR facets are found by equation 3.3. Once the facets are extracted by using the threshold angle, an important step is needed to group the facets into connected surfaces. The extracted facets, shown in figure 4.5, need to be grouped into to connected surfaces. This can be done by using a union-find data structure (also called disjoint-set data structure), which is a data structure to partition a set of elements into a number of subsets without overlapping. The pseudo code of a union-find data structure is demonstrated in algorithm 2.

Algorithm 2 Union-find data structure

```
1:  function MAKESET(x)
2:      x.parent = x
3:  end function
4:  function FIND(x)
5:      if x.parent ≠ x then
6:          x.parent ← Find(x.parent)
7:      end if
8:      return x.parent
9:  end function
10: function UNION(x, y)
11:     xRoot ← Find(x)
12:     yRoot ← Find(y)
13:     if xRoot ≠ yRoot then
14:         yRoot.parent ← xRoot
15:     end if
16: end function
```



(a) multiple VTRs are in the slicing direction



(b) 2D demonstration of VTRs and UTRs in the model

Figure 4.6: Demonstration of the sub-case 1.

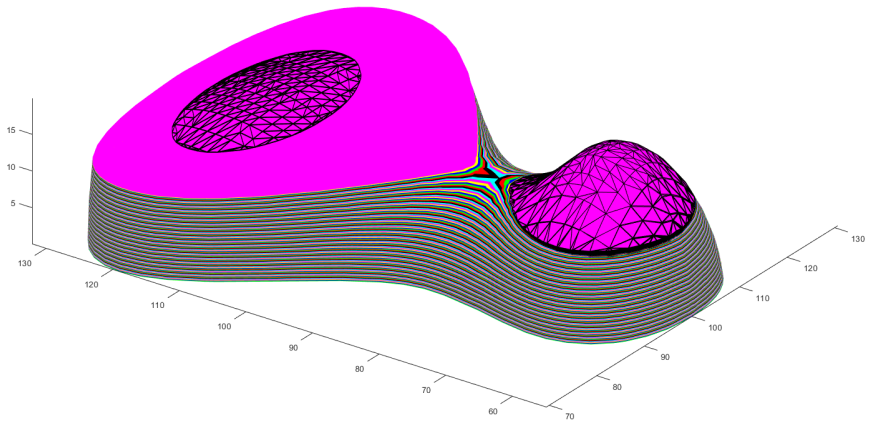
The grouping procedure is performed as follows: First, each facet in F_{vtr} is stored in the data structure using $MakeSet$ function. Then, the system iterates through the set; if a facet is connected to any other present facet, i.e. the two facets share two vertices, the union of this facet with the connected facet is obtained using the $Union$ function. After this procedure, all the facets are grouped into sub-sets where the connected facets are stored.

Once the extracted facets are grouped, the z values of the dividing planes for each group of facets can be determined by using the same procedure described in section 3.2.1. Then, the model can be rebuilt as present in section 3.2.2. Lastly, the layers are generated through a procedure similar to the section 3.3. Note that in section 3.3, for each layer, the UTR is determined by using a polygon binary operation, shown in equation 3.11. In this case, the P_{vtr} varies with different layers, as each VTR occupies a different range of the layers. In the example used in this sub-case, the higher layers only have VTR 1 and the lower layers have both VTR 1 and VTR 2. Therefore, the range of each VTR needs to be stored; when generating the layers, this information is recalled to determine the P_{utr}.

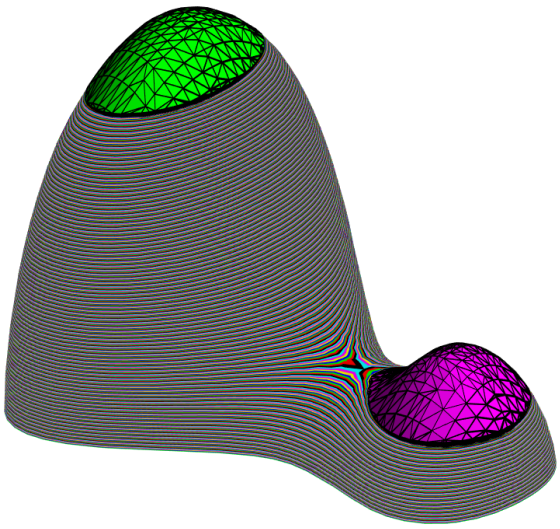
The slicing results of the example model are shown in figure 4.7. Figure 4.7a shows the layers that are occupied by VTR 2 and figure 4.7b presents all the layers.

4.4 Limitations

The more generalized slicing method proposed in this chapter addresses some of the limitations identified from the method presented in chapter 3. The two sub-cases address multiple VTRs in the slicing direction and orthogonal slicing direction. As shown in the two sub-cases, it can be proved that a model can be sliced using the proposed procedure if the model has multiple VTRs with different numbers of layers and curved top and bottom surfaces. However, this method still has limitations. The model must not need support structures that are placed on or inside the model, i.e if the model needs a support structure, the support structure must touch the build plate. Otherwise, the method presented in this chapter is general towards shapes that do not adhere to this limitation.



(a) Generated layers for the bottom part of the model



(b) Generated layers of the model

Figure 4.7: The slicing result of the example model of the sub-case 1.

CHAPTER 5
STUDY OF EFFECTS OF PROCESS PARAMETER ON MATERIAL'S PROPERTIES

Experimental evaluation of the variable layer thickness slicing algorithm was accomplished using two metrics. Each metric was assessed under different process parameters, i.e., top surface slope angles or curvatures, infill patterns, etc. The performances were compared between the variable layer thickness slicing algorithm and the uniform slicing algorithm. The metrics that were measured are as follows:

1. **Top Surface Integrity:** This metric compares the top surface roughness and waviness between the top surfaces of the parts using uniform slicing and variable layer thickness slicing under different geometrical parameters and infill angle.

2. **Tensile Strength:** This metric assess the tensile strengths of the parts printed using uniform slicing and variable layer thickness slicing.

In this chapter, the experimental data are presented, analyzed and discussed.

5.1 Top Surface Integrity

In AM, the surfaces of the object are approximated by a set of triangles, which causes chord error [72]. In addition, the stair-step effect results in unsmooth surfaces. Specifically, in the case of FDM, road width, layer thickness, and extrusion temperature contribute to poor surface finish of AM parts [26, 72–74].

This section evaluates the surface integrity of the top surface under three angles of slopes, i.e. 25°, 5°,10°. For each slope angle, the parts are printed by both variable layer thickness slicing and uniform slicing algorithms. Specifically, for the variable layer thick-

ness slicing method, the roughness of two infill patterns are measured, i.e. the infill lines that are parallel and perpendicular to the measuring path.

5.1.1 Specimens

The specimens are sloped objects printed with variable layer thickness slicing and uniform slicing algorithms. Figure 5.1 demonstrates the infill line and measuring line directions. θ is the slope angle in this drawing.

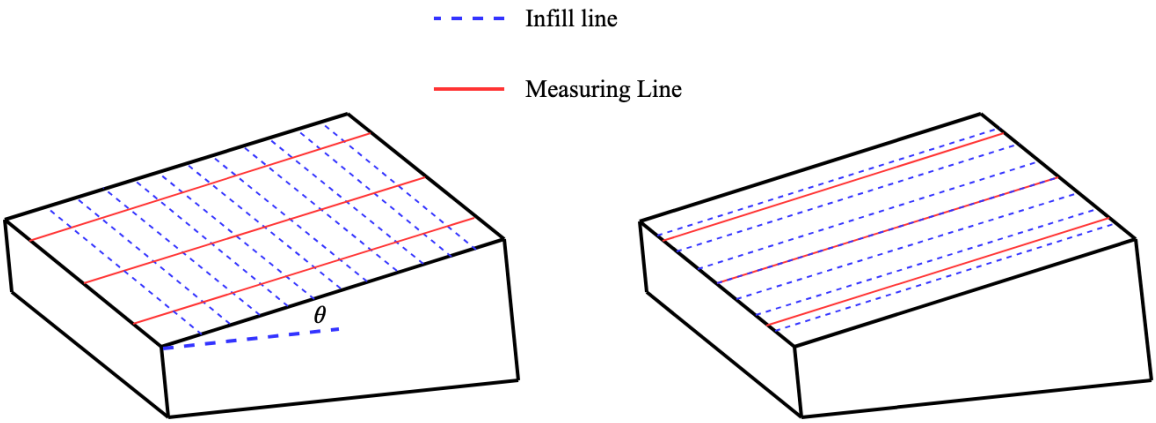


Figure 5.1: Demonstration of specimens with infill lines and measuring lines.

5.1.2 Test Equipment

Profilometer

The profilometer used in this research to quantify the top surface roughness and waviness is a SURFTEST SJ-410 manufactured by Mitutoyo company. Figure 5.2 shows the SJ-410 profilometer.

Some important specifications of this profilometer are given in table 5.1. In this research, Gaussian filter is chosen to filter the raw profile data, and the cut-off length is set to 8mm. The range of each measurement is set to 40mm, which covers 4/5 length of the surface length of the slopes.

Figure 5.2: Mitutoyo SURFTEST SJ-410 profilometer [75].

Table 5.1: Mitutoyo SJ-410 specifications.

Metric		Unit	Value
Measuring range		mm	50
Measuring force		mN	4
Measuring speed		mm/s	0.05, 0.1, 0.2, 0.5, 1.0
Filter	Type	-	2CR, PC75, Gaussian
	Cut-off length	mm	0.08, 0.25, 0.8, 2.5, 8.0

5.1.3 Results

Surface roughness

The measurements from the profilometer are presented in this section. The result is then analyzed and discussed. As shown in figure 5.1, for each slope specimen, three measurements are performed along three measuring paths on the top surface.

A number of parameters can be used to evaluate surface roughness. Ra, known as the arithmetic average value, and Rq, known as the root mean squared, are the most commonly used parameters to calculate one-dimensional roughness. Ra is determined from the arith-

metic average of the absolute values of the profile height deviations from the mean line:

$$Ra = \frac{1}{n}\sum_{i=1}^{n}|y_i - \overline{y}| \tag{5.1}$$

$$\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{5.2}$$

where y_i is the filtered profile height, while Rq is calculated as the root mean square of the values of the profile height deviations from the mean line:

$$Rq = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^2} \tag{5.3}$$

$$\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{5.4}$$

where y_i is the filtered profile height.

Rq is chosen in this research to evaluate the surface roughness of the parts due to its good sensitivity to large peeks and valleys. Theoretically, the largest valleys happen at the corners between each layer, which is equal to the cusp height, but the stylus of the profilometer has some interference with the layered part, as shown in figure 5.3. The actual largest valleys occur when the upper corner of each layer touched the side of the stylus. The theoretical largest valley values (cusp height) versus slopes, under different layer thickness, are calculated and presented in figure 5.4. Figure 5.4a demonstrates the theoretical cusp height, while figure 5.4b considers the interference of the stair steps and the profilometer stylus represented in figure 5.3. Since the angle of the stylus is 45°, the stylus would not interfere with the part surface if the slope angle were greater than 45°.

Another parameter that is considered in this research is Rz, which is calculated by the average of the five highest peeks and lowest valleys over the entire sampling length:

$$Rz = \frac{1}{5}\sum_{i=1}^{5} R_{p_i} - R_{v_i} \tag{5.5}$$

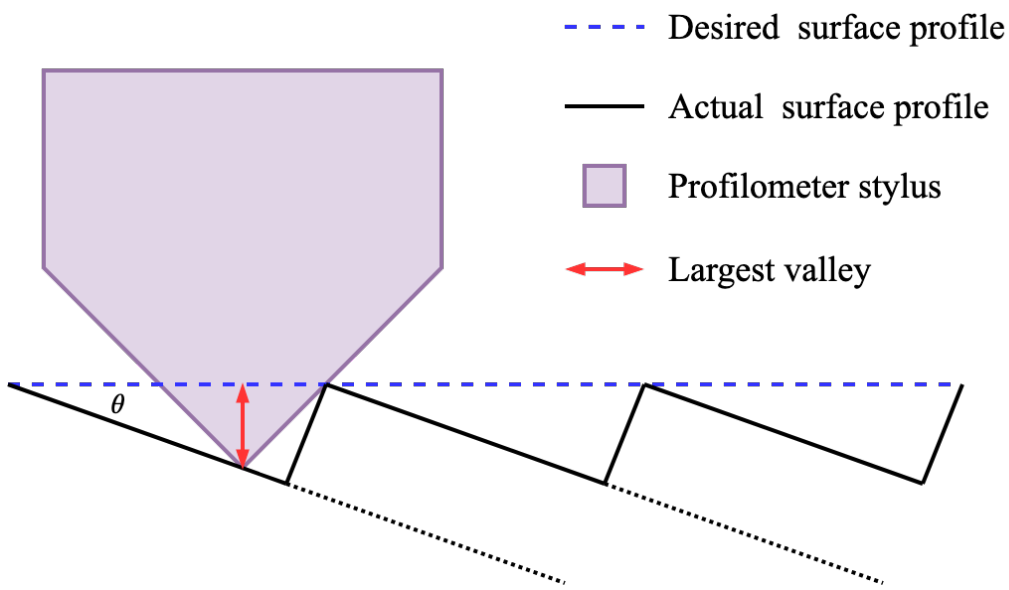
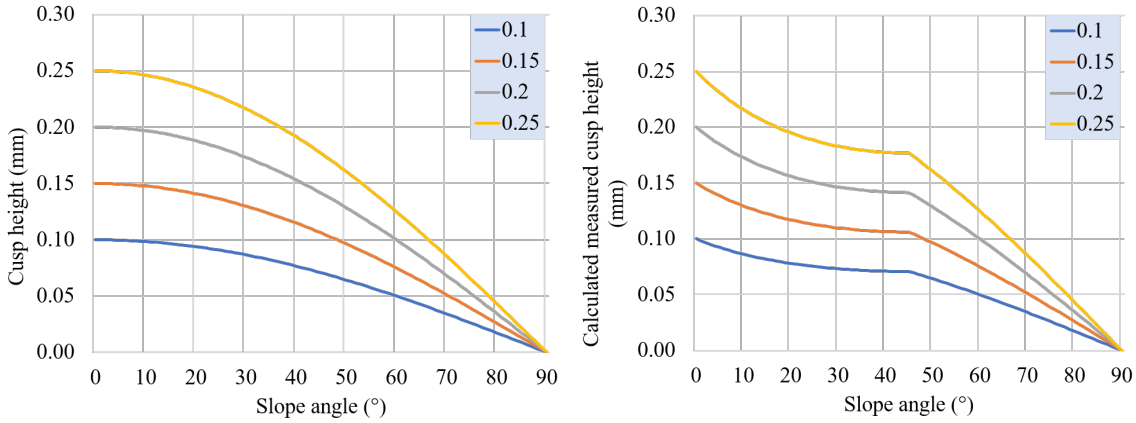


Figure 5.3: The interference between profilometer stylus and part surface.

where R_{p_i} and R_{v_i} are the ith highest peak and lowest valley respectively.

The profile height data was recorded for every 5μm along the measuring path; thus, there are 8000 samples for each measuring path with the length of 40mm. Three cases are measured for each degree of slope: case 1 uses uniform slicing algorithm, case 2 and case 3 use variable layer thickness slicing, with infill lines perpendicular and parallel to the measuring lines correspondingly. The measured profiles of the 2° slope are shown in figure 5.5, and the profiles of the 5° slope and the 10° slope are presented in figure 5.6 and figure 5.7 accordingly.

The profile height deviation along the measuring line, which is case 3, is relatively random when the infill lines are parallel to the measuring line. This is because the deviation along an infill line is caused by the limit of the resolution of the 3D printer and the process parameters, e.g. the extrusion condition, environment temperature, etc. On the other hand, compared to case 3, the profile height deviation is larger in case 2, where the infill lines are perpendicular to the measuring line. In addition, the pattern of the profile height deviation reveals the infill lines in the case 2. Case 1 has the largest deviation among the three cases due to the stair-step effect. Similar to case 3, the pattern of the deviation implies the length



(a) Cusp height

(b) Cusp height with stylus geometry

Figure 5.4: The cusp height versus slope angle under different layer thickness.

of the stairs. The waviness of the profiles will be analyzed later in this section. Figure 5.8 illustrates the statistics of the Rq of the three cases. The error bars in this figure indicate the standard deviations per degree per case.

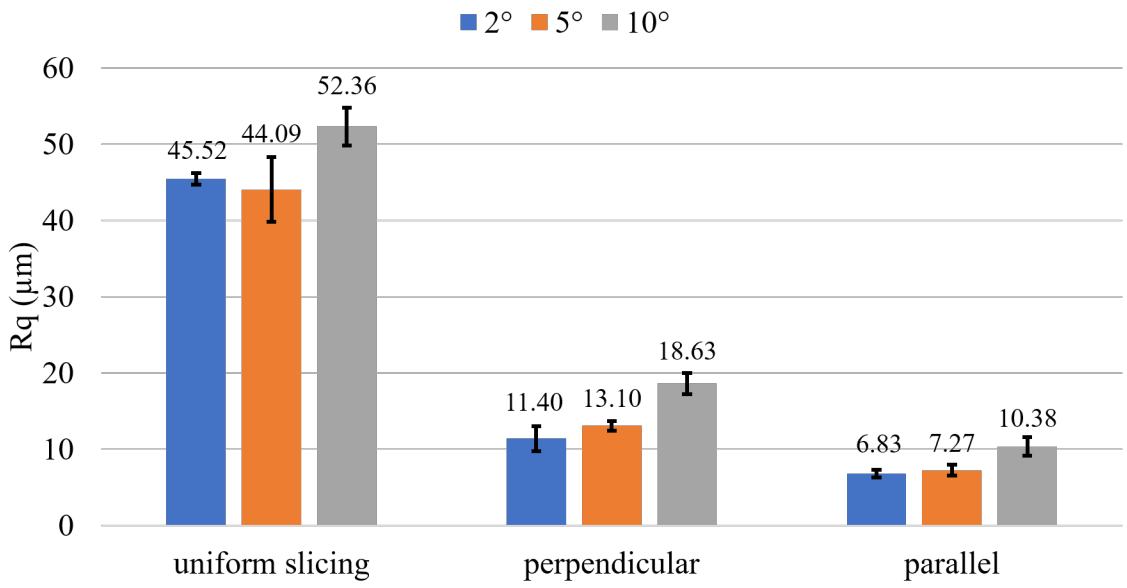
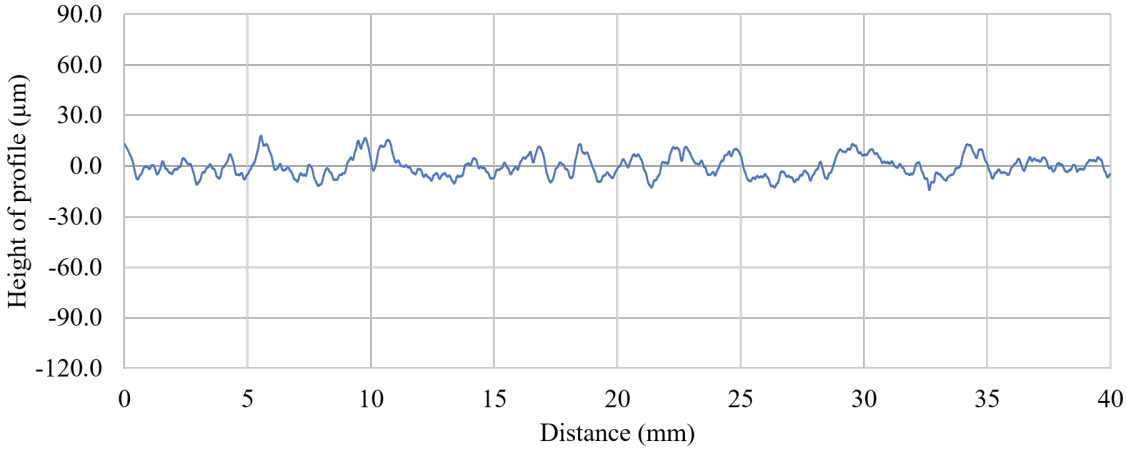
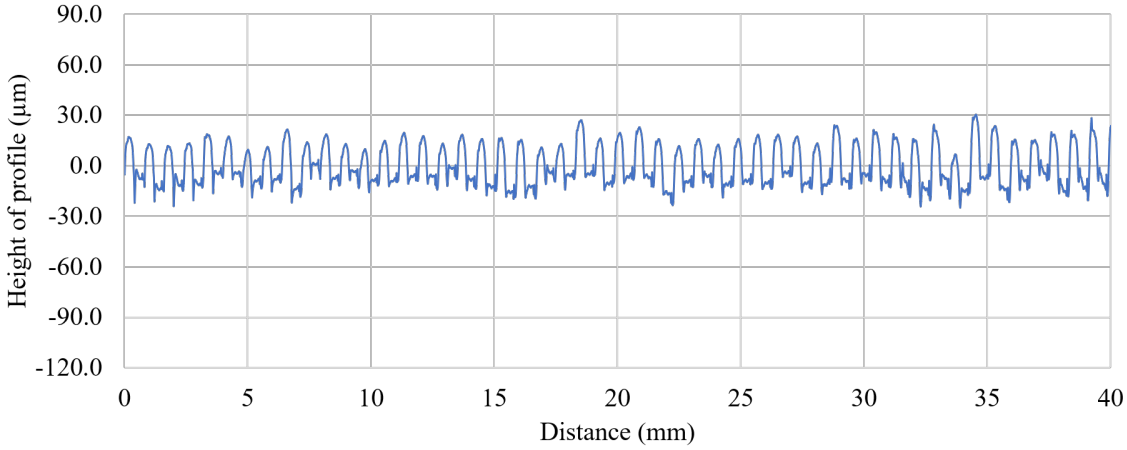


Figure 5.8: The statistic of Rq measured from three cases.

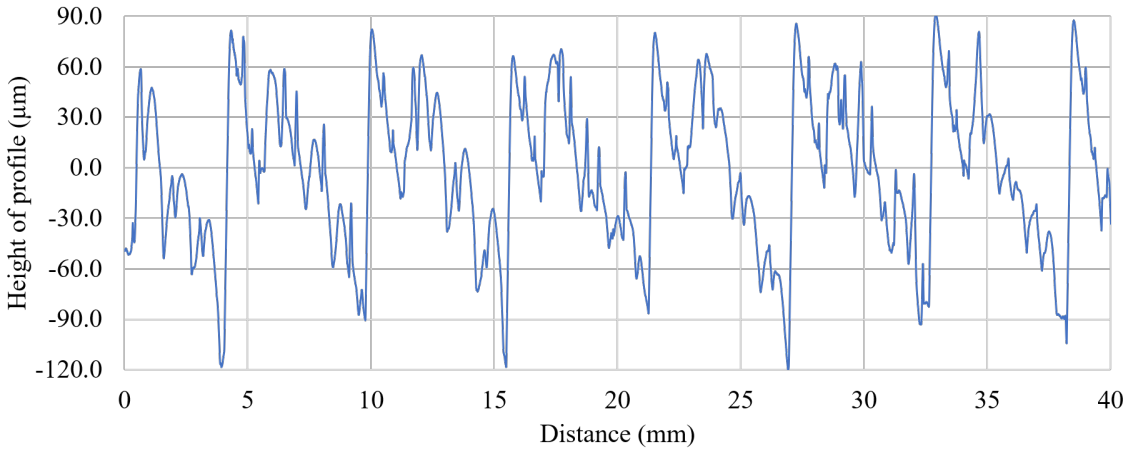
To ensure that the average Rq of each case per degree are different, a t-test is used to evaluate the significance level between each cases. Table 5.2 shows the results of the t-test for each case. The p-values are all smaller than the chosen statistical significance $\alpha = 0.05$,



(a) Infill lines are parallel to the measuring line

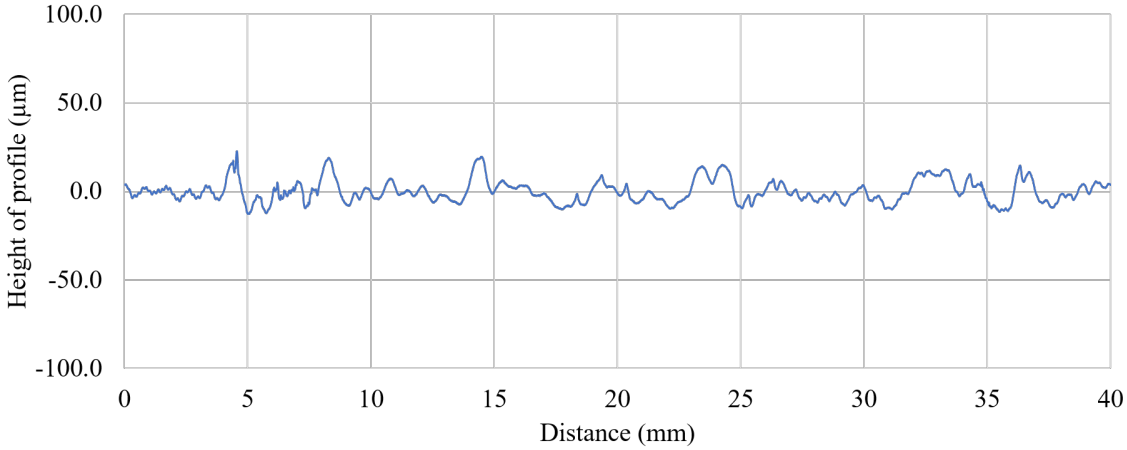


(b) Infill lines are perpendicular to the measuring line

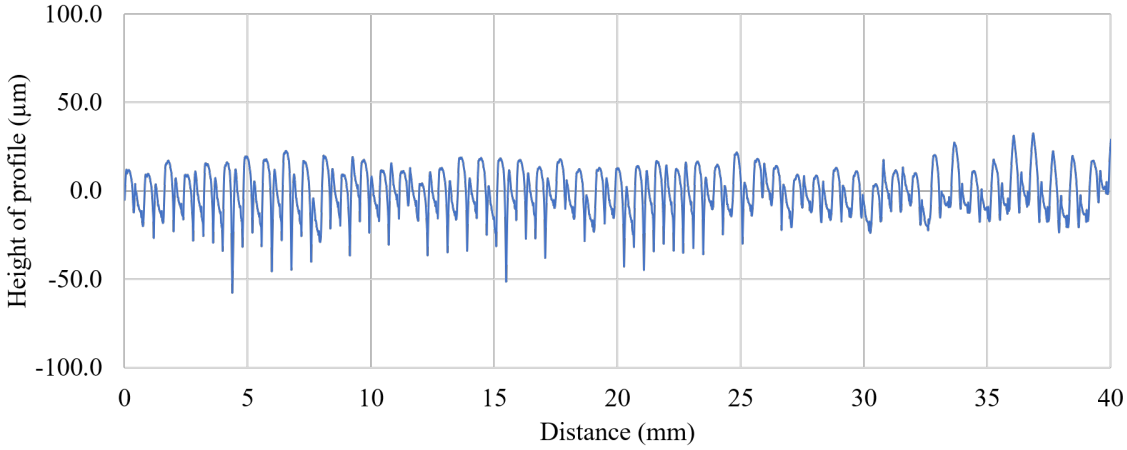


(c) Uniform slicing

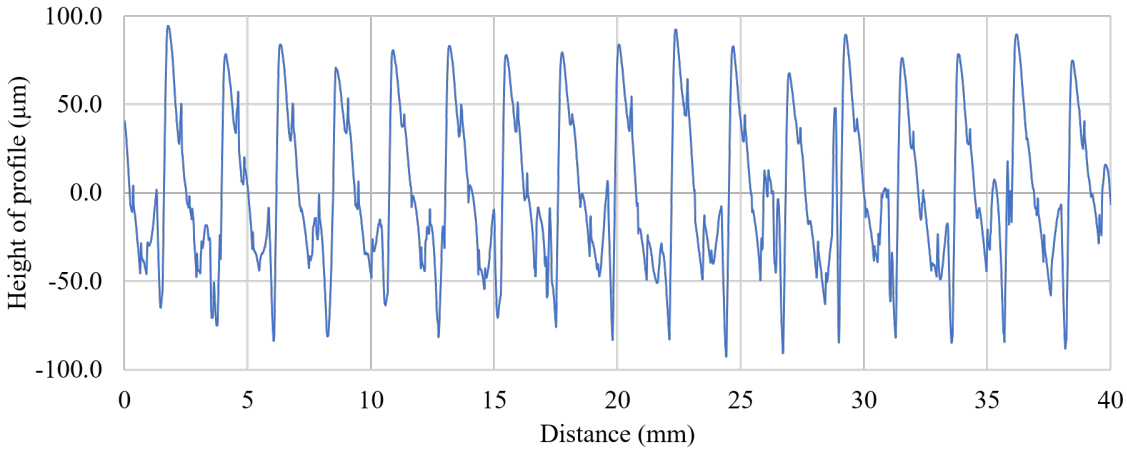
Figure 5.5: Measured profiles of the 2° slope.



(a) Infill lines are parallel to the measuring line

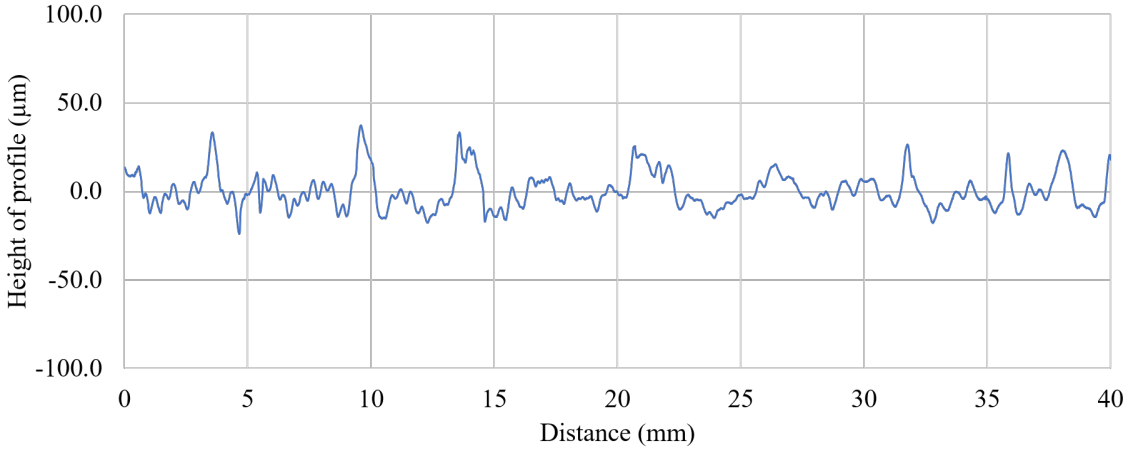


(b) Infill lines are perpendicular to the measuring line

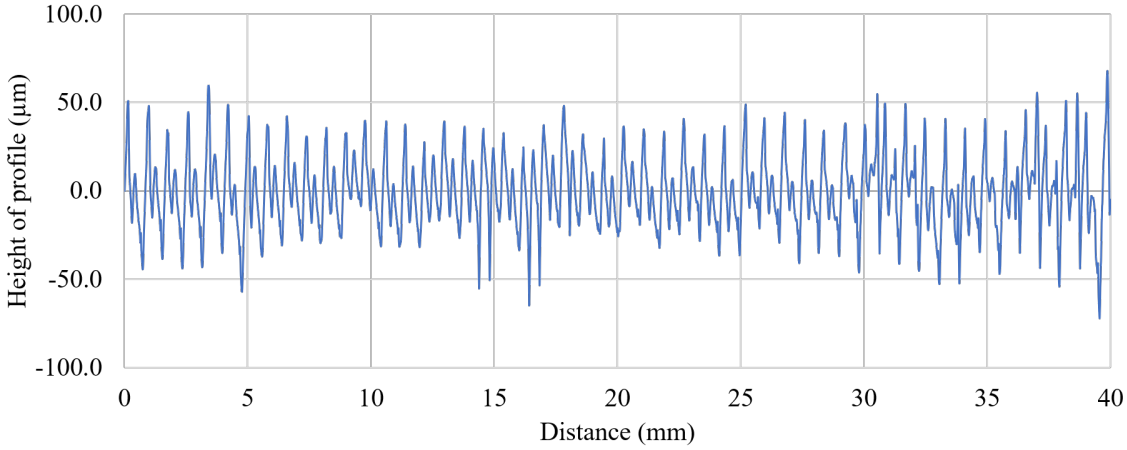


(c) Uniform slicing

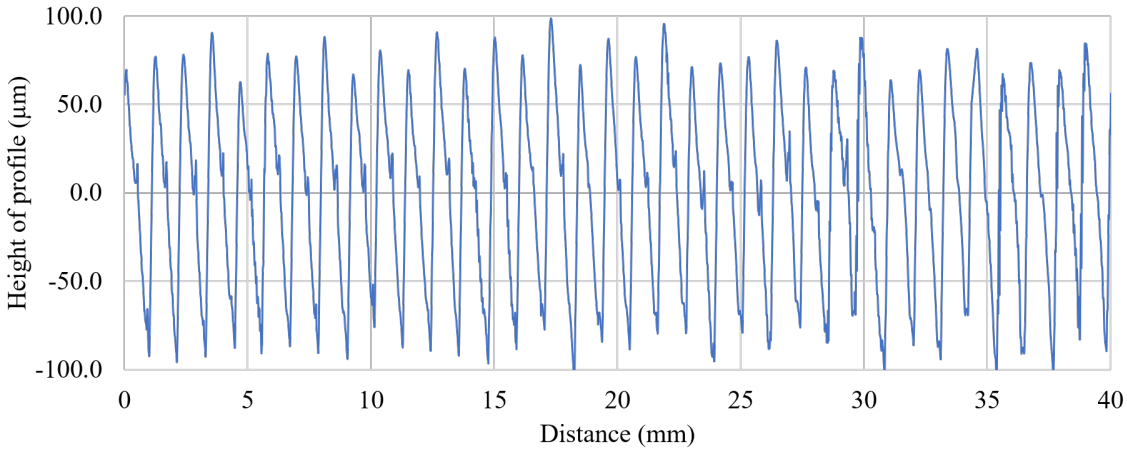
Figure 5.6: Measured profiles of the 5° slope.



(a) Infill lines are parallel to the measuring line



(b) Infill lines are perpendicular to the measuring line



(c) Uniform slicing

Figure 5.7: Measured profiles of the 10° slope.

which indicates that the Rq of case 1 is significantly greater than that of case 2 and case 3. And Rq of case 2 is greater than that of case 3.

Table 5.2: T-test between the Rq of each case.

	perpendicular			parallel		
	2°	5°	10°	2°	5°	10°
uniform slicing	1.97E-04	8.14E-03	3.74E-04	1.95E-06	5.41E-03	2.89E-04
perpendicular	-	-	-	5.00E-02	1.27E-03	3.22E-03

The same analysis is performed on Rz. Figure 5.9 demonstrates the statistics of the Rz of the three cases. The error bars in this figure indicate the standard deviations per degree per case. Table 5.3 shows the results of the t-test between each case. The p-values are all

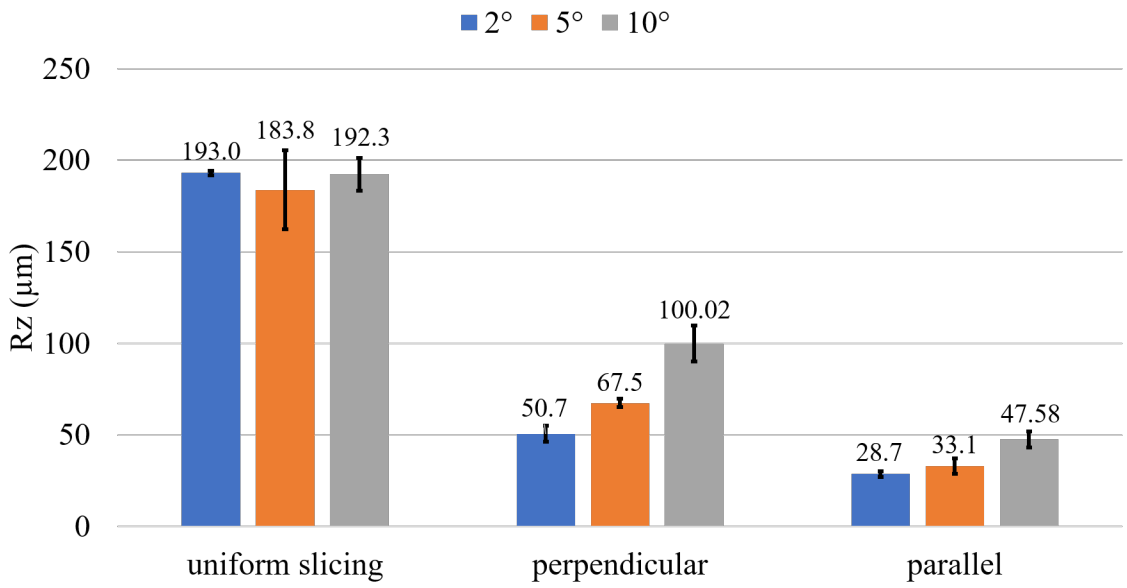


Figure 5.9: The statistic of Rz measured from three cases.

smaller than the chosen statistical significance $\alpha = 0.05$, which indicates that the Rq of case 1 is significantly greater than that of case 2 and case 3. And Rq of case 2 is greater than that of case 3.

As shown in figure 5.4, the cusp height decreases as the slope angle increases. This trend is more significant when considering the stylus geometry, as indicated in figure 5.4b. However, this trend is not seen in the measured data shown in figure 5.8. This is because

Table 5.3: T-test between the Rz of each case.

	perpendicular			parallel		
	2°	5°	10°	2°	5°	10°
uniform slicing	2.07E-04	1.59E-02	6.30E-04	9.02E-08	8.18E-03	2.91E-04
perpendicular	-	-	-	1.22E-02	1.88E-03	7.77E-03

the difference of the cusp height with different slope angles measured in this research is small relative to the standard deviation of Rq in each case. Table 5.4 specifies the cusp heights under the slope angles that were tested in this research. The layer thickness is set to 0.2mm.

Table 5.4: The difference of cusp height under different slope angles.

	2°	5°	10°
Cusp height (mm)	1.999E-01	1.992E-01	1.970E-01
Diff (μm)	-	6.392E-01	2.277E+00
Cusp height considering stylus geometry (mm)	1.934E-01	1.846E-01	1.726E-01
Diff (μm)	-	8.757E+00	1.197E+01

Note that the difference of theoretical cusp height between slopes is in the order of μm, which is smaller than the standard deviation of Rq; thus it is not noticeable from the measured Rq values. When considering the stylus geometry, the differences of the cusp height are in the order of 10μm, which should be picked up by Rq. However, the layer edge is considered square in this model. The upper corner of the layer edge pushes the stylus away from the lower corner of the layer edge, but, in reality, the layer edge is closer to an ellipse than to a square. Thus, the actual difference of the cusp height between different slope angles is much smaller than the ones calculated by the squared layer edge model.

Table 5.5 shows the comparison of Rq and Rz between uniform slicing and variable layer thickness slicing. The Rq is reduced by 76.2% and the Rz is reduced by 76.2%. The surface roughness is also dependent on the direction of the extrusion paths. The Rq and Rz

Table 5.5: Comparison of Rq and Rz between uniform slicing and variable layer thickness slicing.

	uniform slicing (μm)	perpendicular (μm)	parallel (μm)
Rq	47.32	14.38	8.16
		11.27	
Rz	189.74	54.60	36.46
		45.53	

are smaller when the measuring paths and the extrusion paths are parallel.

Surface waviness

Waviness is another topography characteristic of surface integrity; waviness measures the spaced components of surface texture. AW, known as mean spacing of waviness motifs, can be calculated by the arithmetical mean value of the lengths AW_i of waviness motifs, within the evaluation length:

$$AW = \frac{1}{n}\sum_{l=1}^{n} AW_i \tag{5.6}$$

Figure 5.10 demonstrate the waviness profile and AW_i in a measured profile data.

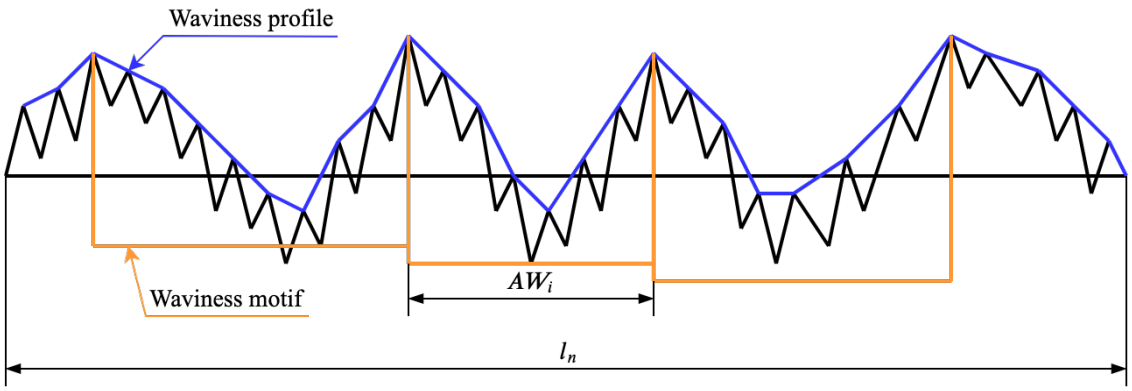


Figure 5.10: A demonstration of waviness profile and AW_i.

To evaluate the waviness of the surfaces, the same measurement data are used to calculate the AW. Instead of finding the peaks of the waviness profile of the data one-by-one, a fast Fourier transform (FFT) is performed on the profile data to extract the frequency information of the data, and then, the average wave spacing can be obtained by looking at

the biggest component in the frequency domain.

The same three cases for each degree of slope that are used in section 5.1.3 are used to measure the surface waviness. Figures 5.12-5.14 present the FFT results of the three cases for each slope angle. The highest bin, which is the dominant frequency, in figure 5.12c, 5.13c, 5.14c implies the inverse of the stair length of each slope angle. Figure 5.11 demonstrates the stair length in relation to the angle of the slope. The stair length decreases quickly when that slope angle is smaller than 10°.

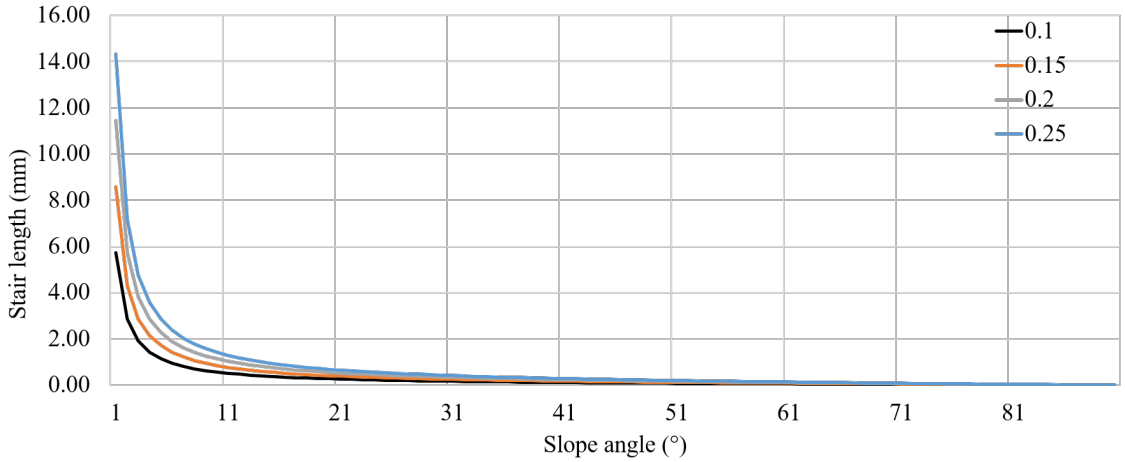
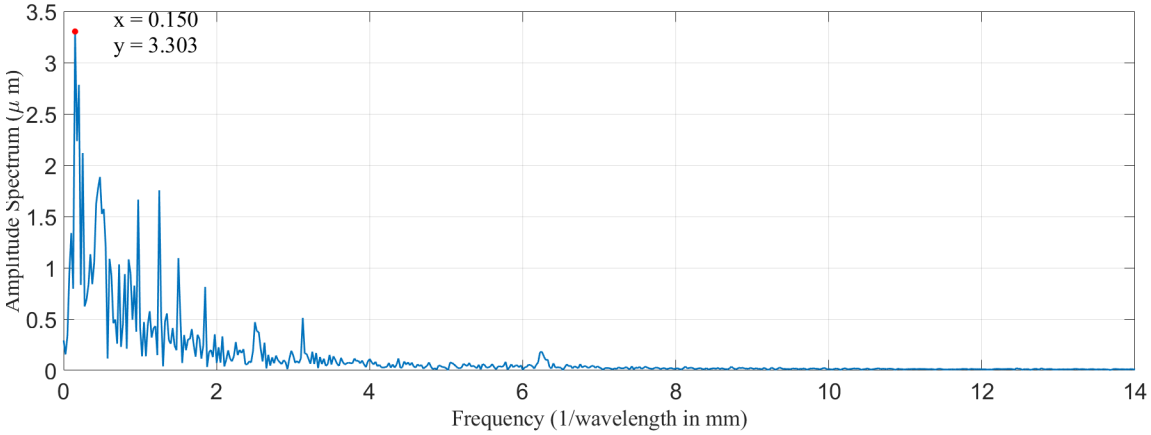
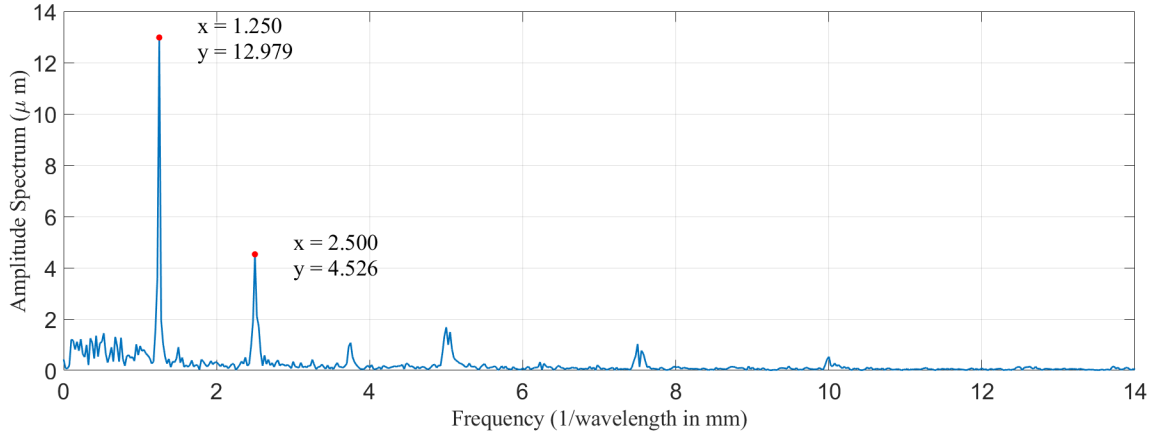


Figure 5.11: The stair length versus slope angle with different layer thicknesses.

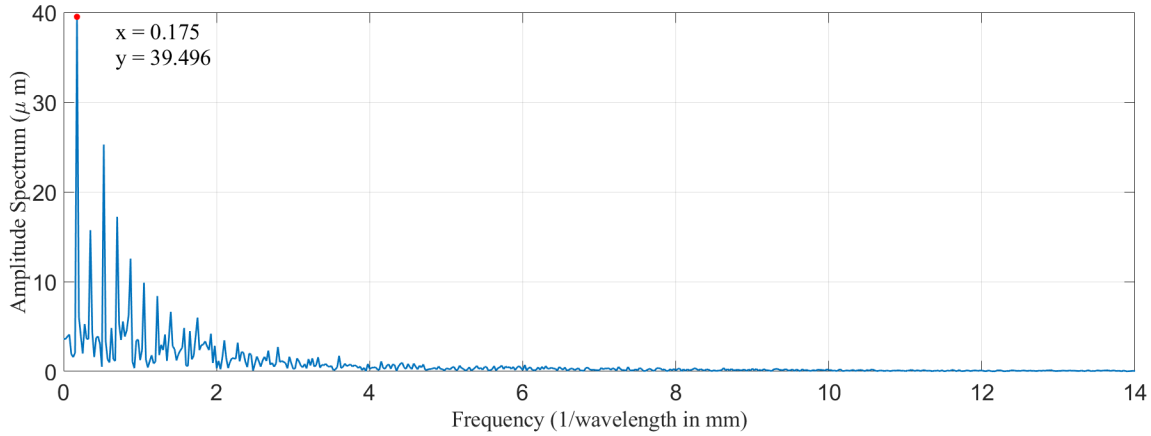
Table 5.6 presents the stair lengths under the three slope angles that are used to measure the profile data in this research. The layer thickness is set to 0.2mm in this table. It can be concluded that the AW values, calculated by the most dominant frequency component, are very close to the theoretical stair length values. This shows that the profile pattern of the stair-step effect can be extracted by calculating the AW value of the profile data.



(a) Infill lines are parallel to the measuring line

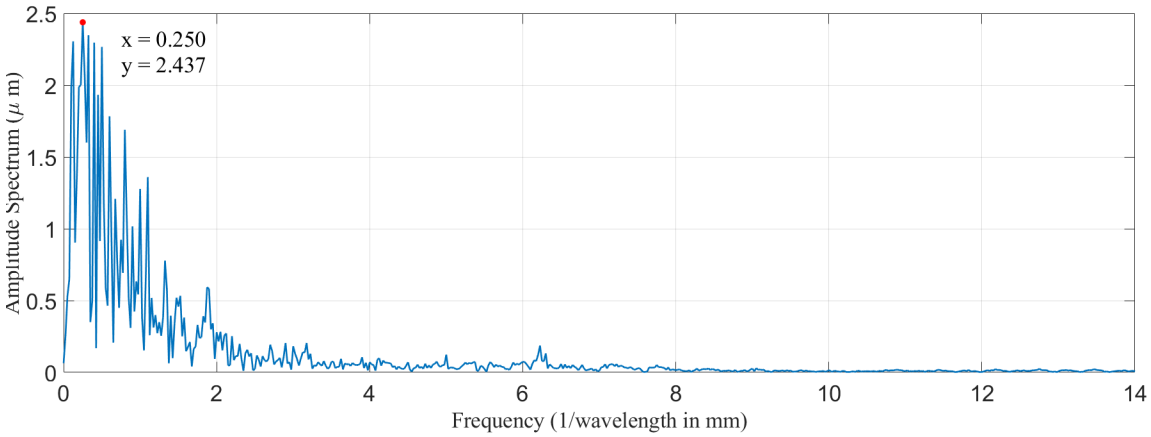


(b) Infill lines are perpendicular to the measuring line

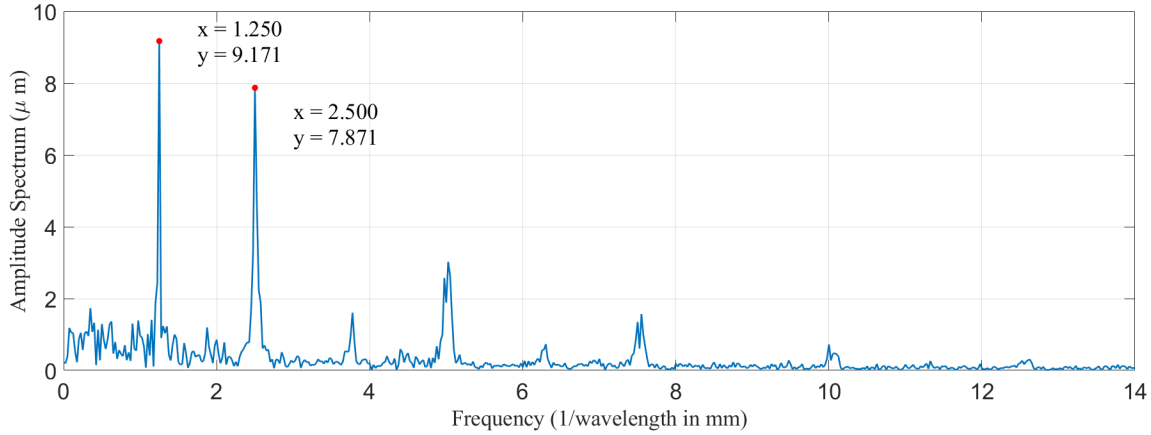


(c) Uniform slicing

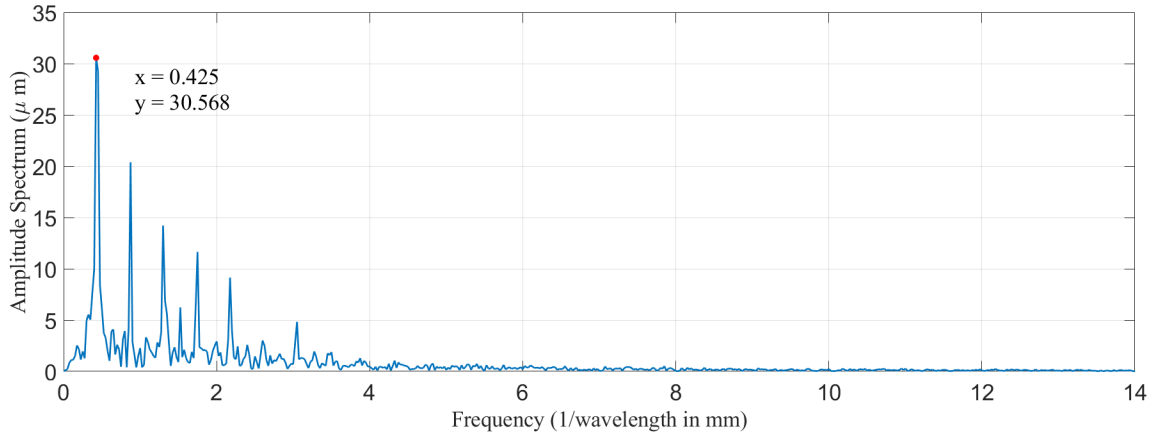
Figure 5.12: FFT spectrum of the 2° slope profile data.



(a) Infill lines are parallel to the measuring line

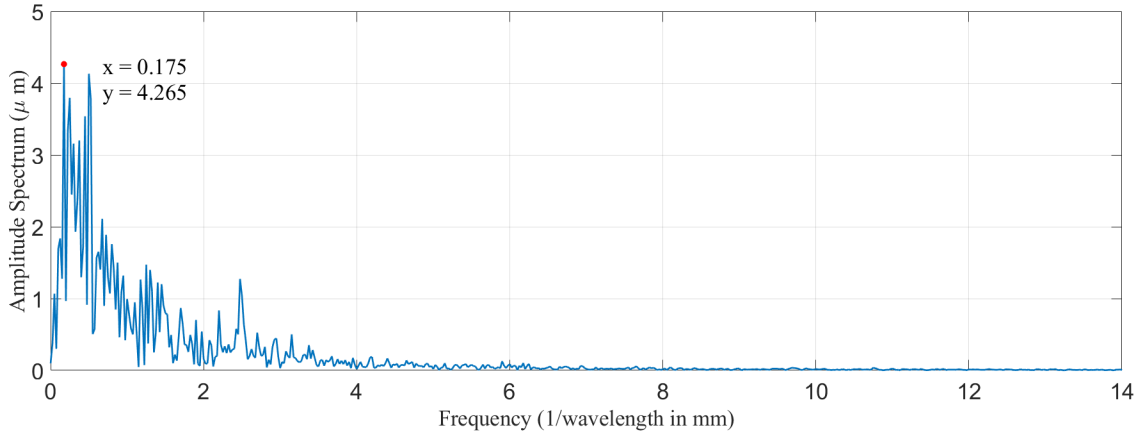


(b) Infill lines are perpendicular to the measuring line

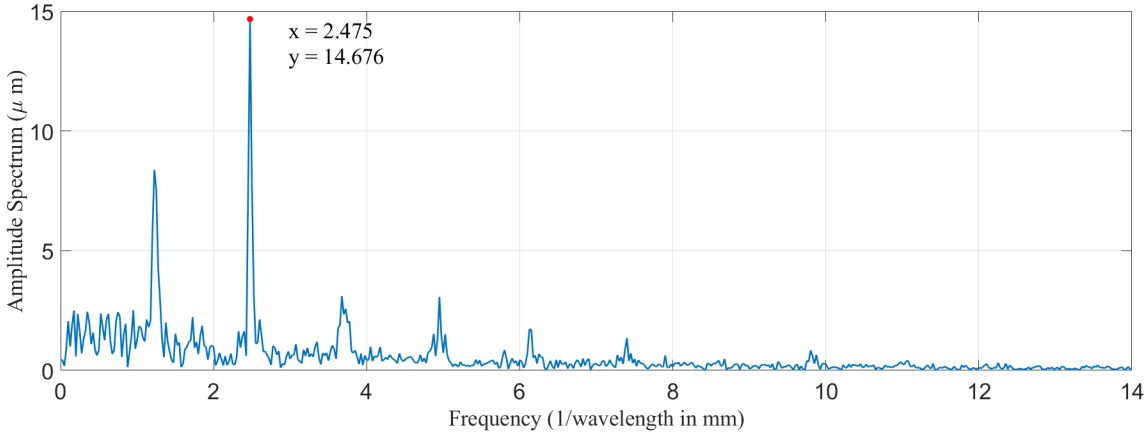


(c) Uniform slicing

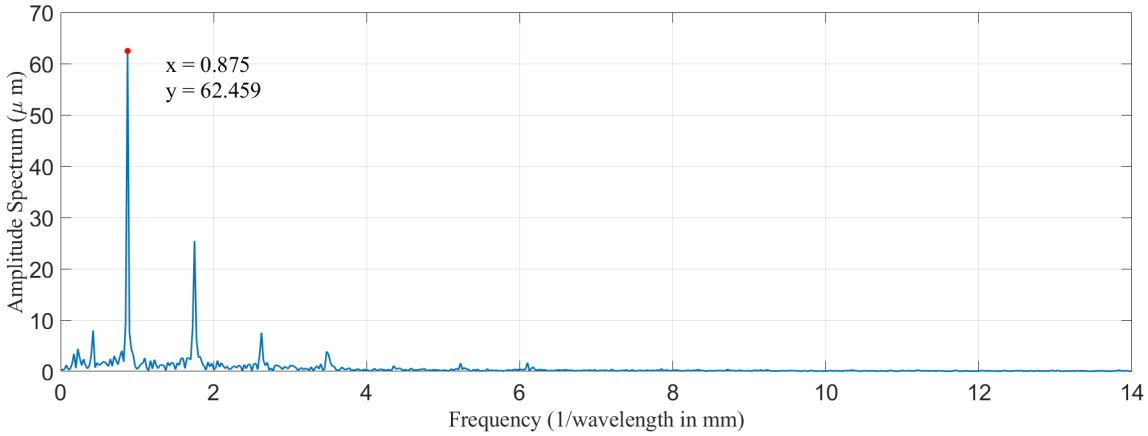
Figure 5.13: FFT spectrum of the 5° slope profile data.



(a) Infill lines are parallel to the measuring line



(b) Infill lines are perpendicular to the measuring line



(c) Uniform slicing

Figure 5.14: FFT spectrum of the 10° slope profile data.

Table 5.6: Stair length under three slope angles with 0.2mm layer thickness.

	$1/AW$	AW (mm)	Stair length (mm)
2°	0.175	5.714	5.731
5°	0.425	2.353	2.295
10°	0.875	1.143	1.152

When the infill lines are perpendicular to the measuring lines, as in case 2, the dominant frequencies in FFT are related to the infill line width. Table 5.7 presents the calculated AW from the most dominant frequency in the FFT of each measured profile data. The AW of 10° slope matches the line width very well, but the AW of the other two slopes is twice the line width. However, the second dominant frequency in the FFT of the 2° and 5° slopes matches the line width perfectly, as shown in figure 5.12b and 5.13b.

Table 5.7: Stair length under three slope angles with 0.2mm layer thickness.

	$1/AW$	AW (mm)	Line width (mm)
2°	1.250	0.8000	0.4002
5°	1.250	0.8000	0.4015
10°	2.475	0.4040	0.4062

On the other hand, the most dominant frequencies in case 3 do not mean much, as the profile data in this case have no apparent pattern. The FFT for this case has a declining curve without any spikes at some specific frequencies.

5.2 Tensile Strength

In the FDM process, each road or layer is considered as a building block of the part. The stack of the road results in anisotropic mechanical characteristics due to variant properties along the road direction and the perpendicular direction [76]. Letcher et al. claimed that

the ultimate tensile strength can vary with different raster orientations in PLA [77]. The internal adhesion is stronger than the inter-road adhesion. Similarly, the stack of layers also leads to anisotropic strength along different directions. One of the advantages of curved layer FDM is higher continuity of filament and better bonding between layers leading to more strength because of larger areas of inter-layer bonding [40]. In a FDM part, there is no continuity of filaments in a cross section, but only flat inter-layers. A tensile force across these is governed by the bonds between layers. Alternatively, in the same part made by curved layer FDM, the tensile force across the layers is partly resisted by the longitudinal tensile strength, which is higher than the bonding strength between adjacent filaments. Other process parameters can also affect the tensile strength of the parts. Kim et al. showed that the printing direction can significantly affect the tensile strength [78].

In this section, the tensile strength of the specimens will be measured. The specimens are built using both uniform slicing and variable layer thickness slicing.

5.2.1 Test Equipment

The tensile test machine, an Instron 5982 100kN load frame, is made by Instron company. Figure 5.15 shows the Instron 5982. This system communicates with the software on the desktop through the controller. The controller receives sensor data and transfers data between the transducers and the computer. The software used to control the system is known as Bluehill 3. Setting parameters, operating the machine, collecting and reporting data is done through the Bluehill software. Some essential controls and some frequently used functions or operations can be done through hardware control of the machine, including starting and stopping a test, controlling the positions through the jog controller, etc.

Figure 5.15: Instron 5982 with 100kN load frame.

5.2.2 Tensile Test Specimen Design

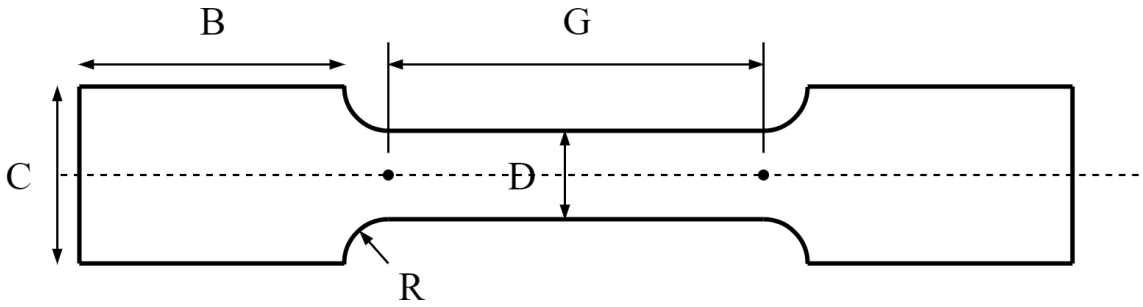


Figure 5.16: Geometry of the round tensile test specimen.

In this research, round tensile test specimens are used based on the consideration of cross sectional area and the specimen printing process. Round specimens can achieve larger cross sectional area with the same gauge width/diameter, which will result in a more accurate tensile strength, considering the geometrical error during printing. The geometry of the specimen is shown in figure 5.16. The dimensions of the specimen are listed in table 5.8.

Table 5.8: Dimensions of the round tensile test specimen.

Dimensions (mm)	
B-Length of grip section	30
C-Diameter of grip section	10
D-Diameter	6
G-Gauge length	36
R-Radius of fillet	2

Instead of machining the specimen from the parts, the specimens tested in this research are printed by both uniform slicing and curved slicing algorithms. The specimen is designed to be a part inside a quadric surface, which is a paraboloid, shown in figure 5.17. The surface paraboloid is described in the following equation:

$$z = -\frac{130}{900} \cdot r^2 + 130 \tag{5.7}$$

$$r = \sqrt{x^2 + y^2} \tag{5.8}$$

The surface of each layer can be described by the equations:

$$z_i = -\frac{130i}{900n} \cdot r^2 + 130\frac{i}{n} \tag{5.9}$$

$$r = \sqrt{x^2 + y^2} \tag{5.10}$$

where z_i is the z value in the ith layer, and n is the total number of layers. By using these equations, the toolpath of the paraboloid model can be obtained using the procedures introduced in chapter 3. For each layer, the paraboloid toolpaths need to be cut into specimen toolpaths, as shown in figure 5.17. Lastly, the G-codes are generated for the specimen.

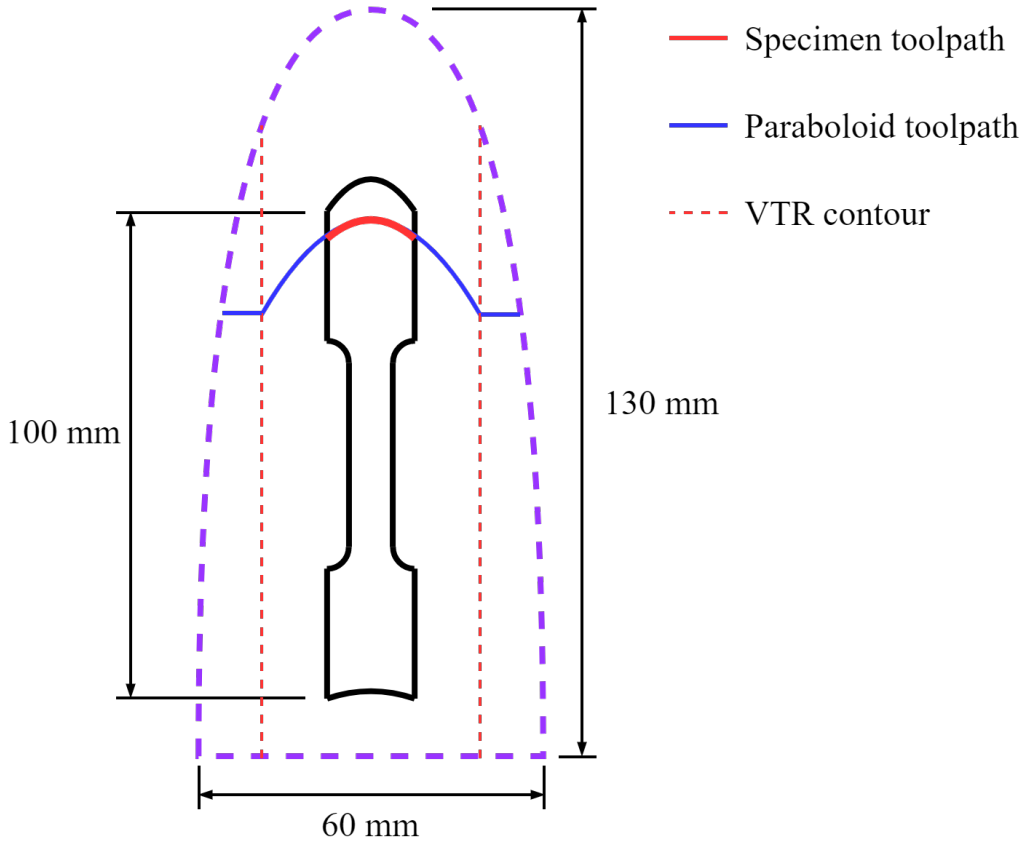


Figure 5.17: The specimen generated from a paraboloid model.

5.2.3 Results

In this test, several specimens are printed and tested for each slicing method, i.e. uniform slicing and variable layer thickness slicing. The final results are based on 5 test for each method. The breaking load is collected from the Bluehill software connected to the tensile machine. In this test, the displacement rate is set to 5mm/min. Table 5.9 presents the ultimate tensile strength in z direction of the specimens made from uniform slicing and variable layer thickness slicing algorithm. The difference in tensile strength depending on the slicing method of the specimen is shown in figure 5.18. The probability associated with the T-test is 0.0196, which is smaller than the significance level 0.05, where $n = 5$.

Some results of the specimens are excluded from the results because the breaking point of the specimen is not located within the gauge length, which is denoted as G in figure 5.16.

The failures are possibly due to the unstable printing environment, variant material properties along the filament, and the precision of the printer. The printing environment, including the nozzle temperature, room temperature and humidity, air flow under the nozzle, etc., was difficult to be precisely controlled during the printing process. These factors affects the final print quality and the layer bonding. The material properties changes over time due to the humidity of the environment [79]. The degradation of PLA plastic introduces the randomness to the strength of the material. Machine precision could be another factor that affects the local strength of the printed specimens. The precision in z direction influence the bonding between layers and the that in $x - y$ direction impacts the bonding between the infill paths.

Table 5.9: Ultimate tensile strength.

Ultimate tensile strength (MPa)		
	Uniform slicing	Variable layer thickness slicing
Average	19.01	21.26
Standard deviation	1.64	1.54
Probability associated with T-test		0.0805
Number of tests per method		5

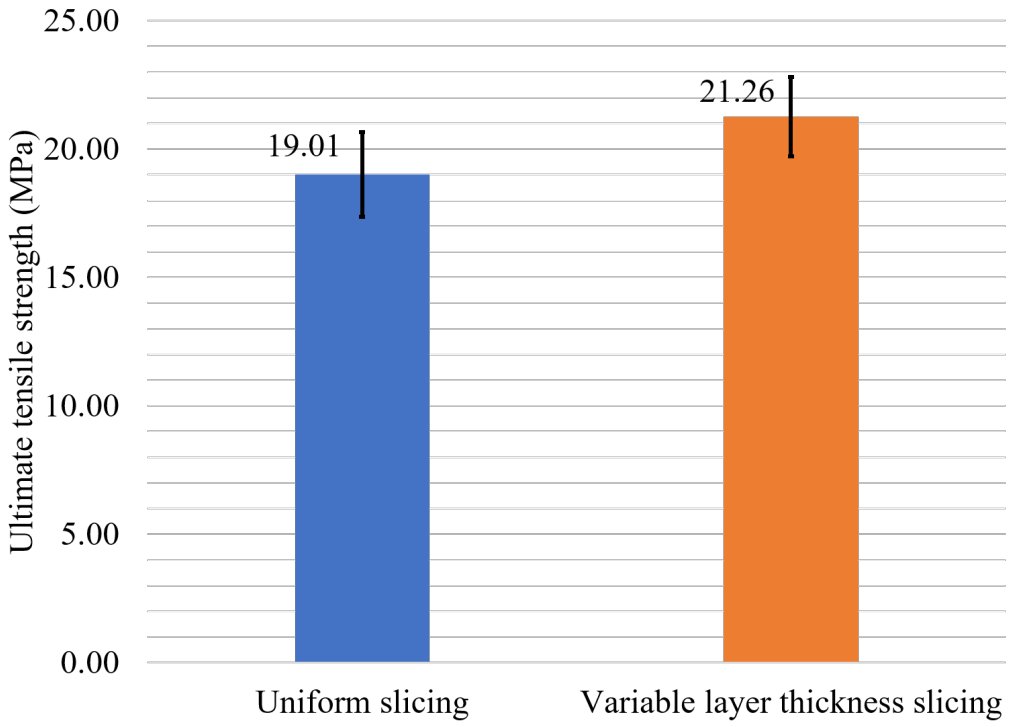


Figure 5.18: Ultimate tensile strength according to the slicing method.

5.3 Summary

This chapter evaluates the variable layer thickness slicing method in surface roughness, surface waviness and ultimate tensile strength. Measurements of surface roughness and waviness are performed using a profilometer. The ultimate tensile strength is evaluated and compared for two slicing methods. Results show that the surface roughness of variable layer thickness slicing is significantly improved over the uniform slicing. The root mean square value, Rq, is reduced by 76.2% and the largest difference value, Rz, is reduced by 71.2%. The ultimate tensile strength in z direction is also improved by 11.8%.

CHAPTER 6
CONCLUSIONS

The objective of the research presented in this dissertation is to develop a general procedure to slice tessellated models for the FDM technology in order to achieve better surface quality and strength. A variable layer thickness slicing procedure is proposed in this thesis. This chapter summarizes the contributions and conclusions of this research and discusses the directions of the future work.

6.1 Contributions

An innovative procedure and algorithm for slicing tessellated models has been presented in this thesis. This procedure addresses the need for eliminating the stair-step effect in conventional slicing procedures. The contributions of this research are listed as below:

- A new variable layer thickness slicing algorithm for base case was proposed and implemented. The stair-step effect is eliminated on the top surface where the slope is close to horizontal. The ultimate tensile strength of the printed objects was improved.
- A slicing procedure that is not limited by the layer thickness ratio was developed to succeed the base case. In this procedure, a models is decomposed into two sub-cases, and then the sub-cases are sliced using the proposed algorithm. This procedure is applicable to more general cases.
- The slicing procedure and G-code generation was implemented on a 3-axis FDM printer and has the potential to be applied on 5-axis FDM printers or robotic FDM.

6.2 Conclusions

The conclusions of this research are itemized below:

- The variable layer thickness slicing procedure for base case addresses the stair-step effect on the surface of manufactured FDM parts. The surface area that suffers the stair-step effect most is the area where the surface slope is close to horizontal. The proposed slicing algorithm eliminates the issue on these areas without sacrificing printing time, i.e. the number of layers.

- The proposed slicing procedure can be applied on both 3-axis and 5-axis FDM printers. The potential of applying the algorithm to 5-axis FDM printers or robotic systems was discussed.

- The surface roughness is significantly improved by moving from the uniform slicing to the variable layer thickness slicing procedure. The root mean square value of the surface roughness, Rq, is reduced by 76.2%. The largest difference value, Rz, is reduced by 71.2%.

- For variable layer thickness slicing, the surface roughness varies in different directions. The Rq of uniaxial profiles that are parallel to extrusion paths is reduced by 43.2% compared to those paths are perpendicular to the extrusion paths, and the Rz is reduced by 33.2%.

- The waviness of the surface can be represented by the dominant frequencies of the profile data. The most dominant frequency in the FFT indicates the average wave spacing (AW) on the surface. For the uniform slicing algorithm, the stair length is revealed by the dominant frequency in the FFT under different surface slopes. While the infill line width appears in the FFT as the dominant or the second dominant frequency when using variable layer thickness slicing, and the infill lines are perpendicular to the measuring paths.

- The tensile strength is weaker in the building direction (z direction) due to the variant properties along the extrusion path (internal-road), and the perpendicular direction

(inter-road). Unlike the uniform slicing, which only has inter-road adhesion on z direction, the variable layer thickness slicing has both inter-road and internal road adhesion on z direction. From the experiments performed in this research, the ultimate tensile strength on z axis is improved by 22.8% when using the variable layer thickness slicing.

6.3 Limitations

The limitations of this research are itemized below:

- The variable layer thickness slicing cannot address the models that need support structure that are placed on or inside the model. The top and bottom surfaces contacted with such support structure will interface with the top and bottom surfaces of the model when calculating the VTR and UTR.
- The bottom surface finish of the part still depends on the build plate surface conditions and support structure surface.
- Even though the proposed slicing procedure can be easily applied to 5-axis machines, the interference issue still needs to be addressed.

6.4 Future Work and Recommendations

The algorithm developed in this dissertation lays the groundwork for future research. There are two directions suggested in this section:

- The variable layer thickness slicing can be applied to a 5-axis FDM machine. This requires both hardware and software development. Specifically, for the hardware, a smaller printing head is needed to avoid interference. This can be challenging because the print head contains a number of essential components, e.g. cooling system, heat sink, sensor, etc. A finer and longer nozzle may also be necessary for better

flexibility of the system. For the software, a trajectory planning algorithm needs to decompose the workpiece in regions that have different building directions, such that the support structure is minimized or the surface quality is optimized. Collision avoidance algorithms are also required, similar to multi-axix CAM system.

- A local slicing algorithm can be applied to the variable layer thickness slicing. For the proposed algorithm in this dissertation, a workpiece is sliced globally, which means the entire workpiece shares the same layer thickness for each layer. This can be unnecessary when the geometry varies significantly over different areas. Thus, local slicing algorithms have the potential to address this issue. It reduces the total print time by using thicker layers in the regions that are not critical for the surface quality. However, this implementation can be challenging subject to complex path planning and interference avoidance requirements.

REFERENCES

[1] I. Gibson, D. W. Rosen, and B. Stucker, *Additive manufacturing technologies*. 2009, pp. 423–436, ISBN: 9781489983640.

[2] B. Evans, *Practical 3D Printers: The Science and Art of 3D Printing*. 2012, p. 579, ISBN: 9781430243922.

[3] P. Mohan Pandey, N. Venkata Reddy, and S. G. Dhande, "Slicing procedures in layered manufacturing: a review," *Rapid Prototyping Journal*, vol. 9, no. 5, pp. 274–288, 2003.

[4] W. Oropallo, L. A. Piegl, P. Rosen, and K. Rajab, "Point cloud slicing for 3-D printing," *Computer-Aided Design and Applications*, vol. 15, no. 1, pp. 90–97, 2018.

[5] A. S. Corp., *Alias Wavefront History*.

[6] "Interchangeable variable block data format for positioning, contouring, and contouring/positioning numerically controlled machines," Electronic Industries Association, 2001 Eye Street, NW, Washington, D.C. 20006, Standard EIA Standard RS-274-D.

[7] K. Lee, "Principles of CAD/CAM/CAE Systems," *New York: Addison-Wesley 1999*, 1999.

[8] R. Scopigno, P. Cignoni, N. Pietroni, M. Callieri, and M. Dellepiane, "Digital Fabrication Techniques for Cultural Heritage: A Survey," *Computer Graphics Forum*, vol. 36, no. 1, pp. 6–21, 2017.

[9] J. Q. Oberhauser, "Design, Construction, Control, and Analysis of Linear Delta Robot," Tech. Rep., 2016.

[10] *3D Printers Explained: Delta, Cartesian, Polar, Scara — All3DP*.

[11] *Polar 3D Launches Unique Polar Coordinate-Based FFF 3D Printer at CES 2015 - 3DPrint.com — The Voice of 3D Printing / Additive Manufacturing*.

[12] H. L. Oo, K. Z. Ye, and Y. H. Linn, "Modeling and controlling of temperature in 3D printer (FDM)," in *Proceedings of the 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, ElConRus 2018*, vol. 2018-January, Institute of Electrical and Electronics Engineers Inc., 2018, pp. 1738–1742, ISBN: 9781538643396.

[13] R. Melnikova, A. Ehrmann, and K. Finsterbusch, "3D printing of textile-based structures by Fused Deposition Modelling (FDM) with different polymer materials," in *IOP Conference Series: Materials Science and Engineering*, vol. 62, Institute of Physics Publishing, 2014.

[14] S. H. Ahn, M. Montero, D. Odell, S. Roundy, and P. K. Wright, "Anisotropic material properties of fused deposition modeling ABS," *Rapid Prototyping Journal*, vol. 8, no. 4, pp. 248–257, 2002.

[15] S. H. Ahn, C. Baek, S. Lee, and I. S. Ahn, "Anisotropic Tensile Failure Model of Rapid Prototyping Parts - Fused Deposition Modeling (FDM)," *International Journal of Modern Physics B*, vol. 17, no. 08n09, pp. 1510–1516, 2003.

[16] L. Novakova-Marcincinova and I. Kuric, "Basic and advanced materials for fused deposition modeling rapid prototyping technology," *Manuf. and Ind. Eng*, vol. 11, no. 1, pp. 24–27, 2012.

[17] J. Kotlinski, "Mechanical properties of commercial rapid prototyping materials," *Rapid Prototyping Journal*, vol. 20, no. 6, pp. 499–510, 2014.

[18] N. Mohan, P. Senthil, S. Vinodh, and N. Jayanth, *A review on composite materials and process parameters optimisation for the fused deposition modelling process*, 2017.

[19] S. Onuh and K. Hon, "Optimising build parameters for improved surface finish in stereolithography," *International Journal of Machine Tools and Manufacture*, vol. 38, no. 4, pp. 329–342, 1998.

[20] A. Dolenc and I. Mäkelä, "Slicing procedures for layered manufacturing techniques," *Computer-Aided Design*, vol. 26, no. 2, pp. 119–126, 1994.

[21] E. Yasa, O. Poyraz, E. U. Solakoglu, G. Akbulut, and S. Oren, "A Study on the Stair Stepping Effect in Direct Metal Laser Sintering of a Nickel-based Superalloy," in *Procedia CIRP*, vol. 45, Elsevier B.V., 2016, pp. 175–178.

[22] C. J. Luis Pérez, J. Vivancos Calvet, and M. A. Sebastián Pérez, "Geometric roughness analysis in solid free-form manufacturing processes," *Journal of Materials Processing Technology*, vol. 119, no. 1-3, pp. 52–57, 2001.

[23] D. Ahn, J. H. Kweon, S. Kwon, J. Song, and S. Lee, "Representation of surface roughness in fused deposition modeling," *Journal of Materials Processing Technology*, vol. 209, no. 15-16, pp. 5593–5600, 2009.

[24] S. in Park and D. W. Rosen, "Quantifying effects of material extrusion additive manufacturing process on mechanical properties of lattice structures using as-fabricated voxel modeling," *Additive Manufacturing*, vol. 12, pp. 265–273, 2016.

[25] A. Boschetto, V. Giordano, and F. Veniali, "Modelling micro geometrical profiles in fused deposition process," in *International Journal of Advanced Manufacturing Technology*, vol. 61, 2012, pp. 945–956.

[26] P. M. Pandey, N. V. Reddy, and S. G. Dhande, "Improvement of surface finish by staircase machining in fused deposition modeling," *Journal of Materials Processing Technology*, vol. 132, no. 1-3, pp. 323–331, 2003.

[27] R. E. Williams and V. L. Melton, "Abrasive flow finishing of stereolithography prototypes," *Rapid Prototyping Journal*, vol. 4, no. 2, pp. 56–67, 1998.

[28] K. F. Leong, C. K. Chua, G. S. Chua, and C. H. Tan, "Abrasive jet deburring of jewellery models built by stereolithography apparatus (SLA)," *Journal of Materials Processing Technology*, vol. 83, no. 1-3, pp. 36–47, 1998.

[29] A. Boschetto and L. Bottini, "Surface improvement of fused deposition modeling parts by barrel finishing," *Rapid Prototyping Journal*, vol. 21, no. 6, pp. 686–696, 2015.

[30] A. Dolenc and I. Mäkelä, "Slicing procedures for layered manufacturing techniques," *Computer-Aided Design*, vol. 26, no. 2, pp. 119–126, 1994.

[31] E. Sabourin, S. A. Houser, and J. H. Bøhn, "Adaptive slicing using stepwise uniform refinement," *Rapid Prototyping Journal*, vol. 2, no. 4, pp. 20–26, 1996.

[32] J. Tyberg and J. H. Bøhn, "Local adaptive slicing," *Rapid Prototyping Journal*, vol. 4, no. 3, pp. 118–127, 1998.

[33] K. Mani, P. Kulkarni, and D. Dutta, "Region-based adaptive slicing," *CAD Computer Aided Design*, vol. 31, no. 5, pp. 317–333, 1999.

[34] R. L. Hope, P. A. Jacobs, and R. N. Roth, "Rapid prototyping with sloping surfaces," *Rapid Prototyping Journal*, vol. 3, no. 1, pp. 12–19, 1997.

[35] W. Ma, W. C. But, and P. He, "NURBS-based adaptive slicing for efficient rapid prototyping," in *CAD Computer Aided Design*, vol. 36, 2004, pp. 1309–1325.

[36] J. Zhang and F. Liou, "Adaptive slicing for a multi-axis laser aided manufacturing process," *Journal of Mechanical Design, Transactions of the ASME*, vol. 126, no. 2, pp. 254–261, 2004.

[37] M. T. Hayasi and B. Asiabanpour, "A new adaptive slicing approach for the fully dense freeform fabrication (FDFF) process," *Journal of Intelligent Manufacturing*, vol. 24, no. 4, pp. 683–694, 2013.

[38] F. Wasserfall, N. Hendrich, and J. Zhang, "Adaptive slicing for the FDM process revisited," in *IEEE International Conference on Automation Science and Engineering*, vol. 2017-August, IEEE Computer Society, 2018, pp. 49–54, ISBN: 9781509067800.

[39] D. A. Klosterman, R. P. Chartoff, N. R. Osborne, G. A. Graves, A. Lightman, G. Han, A. Bezeredi, and S. Rodrigues, "Development of a curved layer LOM process for monolithic ceramics and ceramic matrix composites," *Rapid Prototyping Journal*, vol. 5, no. 2, pp. 61–71, 1999.

[40] D. Chakraborty, B. Aneesh Reddy, and A. Roy Choudhury, "Extruder path generation for Curved Layer Fused Deposition Modeling," *CAD Computer Aided Design*, vol. 40, no. 2, pp. 235–243, 2008.

[41] B. Huang and S. B. Singamneni, "Curved layer adaptive slicing (CLAS) for fused deposition modelling," *Rapid Prototyping Journal*, vol. 21, no. 4, pp. 354–367, 2015.

[42] O. Diegel, S. Singamneni, B. Huang, and I. Gibson, "Curved layer fused deposition modeling in conductive polymer additive manufacturing," in *Advanced Materials Research*, vol. 199-200, 2011, pp. 1984–1987, ISBN: 9783037850374.

[43] R. J. Allen and R. S. Trask, "An experimental demonstration of effective Curved Layer Fused Filament Fabrication utilising a parallel deposition robot," *Additive Manufacturing*, vol. 8, pp. 78–87, 2015.

[44] S. Lim, R. A. Buswell, P. J. Valentine, D. Piker, S. A. Austin, and X. De Kestelier, "Modelling curved-layered printing paths for fabricating large-scale construction components," *Additive Manufacturing*, vol. 12, pp. 216–230, 2016.

[45] W. Sun and P. Lal, *Recent development on computer aided tissue engineering - A review*, 2002.

[46] X. Chen, C. Wang, X. Ye, Y. Xiao, and S. Huang, "Direct slicing from PowerSHAPE models for rapid prototyping," *International Journal of Advanced Manufacturing Technology*, vol. 17, no. 7, pp. 543–547, 2001.

[47] W. Cao and Y. Miyamoto, "Direct Slicing from AutoCAD Solid Models for Rapid Prototyping," *International Journal of Advanced Manufacturing Technology*, vol. 21, no. 10-11, pp. 739–742, 2003.

[48] B. Starly, A. Lau, W. Sun, W. Lau, and T. Bradbury, "Direct slicing of STEP based NURBS models for layered manufacturing," *CAD Computer Aided Design*, vol. 37, no. 4, pp. 387–397, 2005.

[49] W. Oropallo, L. A. Piegl, P. Rosen, and K. Rajab, "Point cloud slicing for 3-D printing," *Computer-Aided Design and Applications*, vol. 15, no. 1, pp. 90–97, 2018.

[50] Z. Zhao and L. Laperrière, "Adaptive direct slicing of the solid model for rapid prototyping," *International Journal of Production Research*, vol. 38, no. 1, pp. 69–83, 2000.

[51] M. Y. Zhou, J. T. Xi, and J. Q. Yan, "Adaptive direct slicing with non-uniform cusp heights for rapid prototyping," *International Journal of Advanced Manufacturing Technology*, vol. 23, no. 1-2, pp. 20–27, 2004.

[52] Y. Sasaki, M. Takezawa, S. Kim, H. Kawaharada, and T. Maekawa, "Adaptive direct slicing of volumetric attribute data represented by trivariate B-spline functions," *International Journal of Advanced Manufacturing Technology*, vol. 91, no. 5-8, pp. 1791–1807, 2017.

[53] S. Sikder, A. Barari, and H. A. Kishawy, "Effect of Adaptive Slicing on Surface Integrity in Additive Manufacturing," in *Volume 1A: 34th Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, 2014, ISBN: 978-0-7918-4628-5.

[54] X. Zheng, K. Cheng, X. Zhou, J. Lin, and X. Jing, "An adaptive direct slicing method based on tilted voxel of two-photon polymerization," *International Journal of Advanced Manufacturing Technology*, vol. 96, no. 1-4, pp. 521–530, 2018.

[55] J. Feng, J. Fu, Z. Lin, C. Shang, and B. Li, "Direct slicing of T-spline surfaces for additive manufacturing," *Rapid Prototyping Journal*, vol. 24, no. 4, pp. 709–721, 2018.

[56] B. R. Vati, "A generic solution to polygon clipping," *Communications of the ACM*, vol. 35, no. 7, pp. 56–64, 1992.

[57] Ultimaker.com, *CuraEngine*.

[58] *ISO - ISO/IEC 21778:2017 - Information technology — The JSON data interchange syntax*.

[59] D. E. Knuth, *The art of computer programming*. Addison-Wesley, 1997, ISBN: 9780321751041.

[60] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*, 1st. MIT Press, 1990, p. 1028, ISBN: 0262031418.

[61] P. K. Agarwal, E. Flato, and D. Halperin, "Polygon decomposition for efficient construction of Minkowski sums," *Computational Geometry*, vol. 21, no. 1-2, pp. 39–61, 2002.

[62] J. D. C. Little, K. G. Murty, D. W. Sweeney, and C. Karel, "An Algorithm for the Traveling Salesman Problem," *Operations Research*, vol. 11, no. 6, pp. 972–989, 1963.

[63] M. Battarra, A. A. Pessoa, A. Subramanian, and E. Uchoa, "Exact algorithms for the traveling salesman problem with draft limits," *European Journal of Operational Research*, vol. 235, no. 1, pp. 115–128, 2014.

[64] J. Glomvik Rakke, M. Christiansen, K. Fagerholt, and G. Laporte, "The Traveling salesman problem with draft limits," *Computers and Operations Research*, vol. 39, no. 9, pp. 2161–2167, 2012.

[65] S. Basu and D. Ghosh, "A Review of the Tabu Search Literature on Traveling Salesman Problems INDIAN INSTITUTE OF MANAGEMENT AHMEDABAD-380015 INDIA A Review of the Tabu Search Literature on Traveling Salesman Problems," Tech. Rep., 2008.

[66] D. E. Knuth, "Postscript about NP-hard problems," *ACM SIGACT News*, vol. 6, no. 2, pp. 15–16, 1974.

[67] H. S. Wilf, "Algorithms and Complexity," 2002.

[68] *All about "layer seams" – Matt's Hub*.

[69] F. Wulle, D. Coupek, F. Schäffner, A. Verl, F. Oberhofer, and T. Maier, "Workpiece and Machine Design in Additive Manufacturing for Multi-Axis Fused Deposition Modeling," in *Procedia CIRP*, vol. 60, Elsevier B.V., 2017, pp. 229–234.

[70] Ø. Kallevik Grutle, "5-axis 3D Printer," Tech. Rep., 2015.

[71] C. Wu, C. Dai, G. Fang, Y. J. Liu, and C. C. Wang, "RoboFDM: A robotic system for support-free fabrication using FDM," in *Proceedings - IEEE International Conference on Robotics and Automation*, Institute of Electrical and Electronics Engineers Inc., 2017, pp. 1175–1180, ISBN: 9781509046331.

[72] D. Ding, Z. Pan, D. Cuiuri, and H. Li, "A practical path planning methodology for wire and arc additive manufacturing of thin-walled structures," *Robotics and Computer-Integrated Manufacturing*, vol. 34, pp. 8–19, 2015.

[73] T. Wohlers, *Future potential of rapid prototyping and manufacturing around the world*, 1995.

[74] J. P. Kruth, M. C. Leu, and T. Nakagawa, "Progress in additive manufacturing and rapid prototyping," *CIRP Annals - Manufacturing Technology*, vol. 47, no. 2, pp. 525–540, 1998.

[75] *Surftest SJ-410 Series 178-Portable Surface*.

[76] Q. Sun, G. M. Rizvi, C. T. Bellehumeur, and P. Gu, "Effect of processing conditions on the bonding quality of FDM polymer filaments," *Rapid Prototyping Journal*, vol. 14, no. 2, pp. 72–80, 2008.

[77] T. Letcher and M. Waytashek, "Material property testing of 3D-printed specimen in pla on an entry-level 3D printer," in *ASME International Mechanical Engineering Congress and Exposition, Proceedings (IMECE)*, vol. 2A, American Society of Mechanical Engineers (ASME), 2014, ISBN: 9780791846438.

[78] H. Kim, E. Park, S. Kim, B. Park, N. Kim, and S. Lee, "Experimental Study on Mechanical Properties of Single- and Dual-material 3D Printed Products," *Procedia Manufacturing*, vol. 10, pp. 887–897, 2017.

[79] K. L. G. Ho, A. L. Pometto, and P. N. Hinz, "Effects of temperature and relative humidity on polylactic acid plastic degradation," *Journal of Environmental Polymer Degradation*, vol. 7, no. 2, pp. 83–92, 1999.